Oracle® Communications Billing and Revenue Management

System Administrator's Guide

Release 12.0
E51029-20
April 2024

Oracle Communications Billing and Revenue Management System Administrator's Guide, Release 12.0

E51029-20

Contents

11 Running Stored Procedures

12 Using Configuration Files

13 Connecting BRM Components

14 Controlling Batch Operations

25 Improving Connection Manager Performance

26 Using Connection Manager Master Process to Improve Performance

27 Improving Data Manager and Queue Manager Performance

Part IV Troubleshooting BRM

37 Resolving Problems in Your BRM System

38 Diagnosing Some Common Problems with BRM

42 Monitoring Business Operations Center

43 Monitoring Billing Care and Billing Care REST API

Part VI Partitioning and Managing BRM Database Tables

44 Partitioning Tables

45 Converting Nonpartitioned Classes to Partitioned Classes

46 About Purging Data

47 Generating Virtual Columns on Event Tables

Part VII Managing a Multischema System

48 Managing a Multischema System

49 Multischema Utilities

59 Setting Up and Managing Elastic Charging Engine Security

60 Starting and Stopping ECE

61 Monitoring ECE Using ECE Monitoring Agent

62 Monitoring ECE Components

63 Configuring Subscriber-Based Tracing and Logging

64 Managing Nodes and Clusters in ECE

65 Configuring ECE Data-Loading Utilities and Data Updaters

76 Troubleshooting ECE

E Diameter Gateway Error Codes

Preface

This guide describes system administration tasks for Oracle Communications Billing and Revenue Management (BRM).

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

Part I

Basic BRM System Administration

This part describes basic Oracle Communications Billing and Revenue Management (BRM) system administration tasks. It contains the following chapters:

1

Starting and Stopping the BRM System

Learn how to start and stop your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

About Starting and Stopping BRM Components

You start and stop BRM components by using the **pin_ctl** utility. You can start and stop individual components, or start and stop multiple components in a specified order.

You can start BRM components as user **root**, user **pin**, or any other name you choose. If you start both the BRM database and the BRM components with a user name other than **root**, you have better control over security and administrators that do not have superuser permissions.

> **Note:**
>
> If you use a port number less than 1000 for a component (1023 for the **cm_proxy** component), you must start that component as the user **root**. If you use a port number greater than 1024, you do not have to start the component as the user **root**.

Order of Starting and Stopping Components

You must start and stop BRM components in a specific order. Start the database first and start client applications last. You stop BRM components in the opposite order.

To start multiple components:

1. Start the BRM database. The BRM database usually starts automatically when the computer is started.
2. Start the Data Manager (DM) (**dm_oracle**).

> **Note:**
>
> In multischema systems, you must start all secondary DMs before you start the primary DM.

3. Start any other DMs, such as **dm_fusa** and **dm_vertex**.
4. Start the daemon or process for any optional features such as tMailer and Popper.
5. Start the Connection Manager Master Processes (CMMPs) if your system uses them.
6. Start CM Proxy if your system uses it.
7. Start the CMs.
8. Start BRM clients and other programs, such as service integration components.

Components Monitored and Controlled by the pin_ctl Utility

You can use the **pin_ctl** utility to start and stop the following BRM components:

- Connection Manager
- CM Master Process (CMMP)
- Connection Manager Proxy (cm_proxy)
- Data Managers:
  - Oracle Data Manager
  - Email Data Manager
  - EAI Data Manager
  - Paymentech Data Manager
  - Invoice Data Manager
- EAI Java Server
- Invoice Formatter
- Paymentech Answer Simulator
- Batch Controller

You can configure the **pin_ctrl** utility to start and stop other components. See "Starting and Stopping Optional Components by Using pin_ctl".

Running the pin_ctl Utility

To start a BRM component by using the **pin_ctl** utility:

1. Go to the *BRM_home*/**bin** directory.
2. Run the **pin_ctl** utility with the **start** action:

   ```
   pin_ctl start component
   ```

   where *component* is the component you want to start. For example, to start the Oracle DM:

```
pin_ctl start dm_oracle
```

For a list of component names, see "Parameters for Components". To use the **pin_ctl** utility to start and stop a component that is not included by default, see "pin_ctl".

You have the following options for running the **pin_ctl** utility:

- Get the status of a component:

  ```
  pin_ctl status component
  ```

- Start a component and clear its log files at the same time:

  ```
  pin_ctl cstart component
  ```

- Clear the log files for a component:

  ```
  pin_ctl clear component
  ```

- Get get diagnostic data when starting a component, or getting the status:

  ```
  pin_ctl status -collectdata component
  ```

- Halt a component by running the **kill -9** command:

  ```
  pin_ctl halt component
  ```

- Halt and restart a component:

  ```
  pin_ctl restart component
  ```

- Stop a component:

  ```
  pin_ctl stop component
  ```

Setting Up and Configuring the pin_ctl Utility

Install the **pin_ctl** utility executable on any system that runs a BRM component.

Each instance of the **pin_ctl** utility is configured by a **pin_ctl.conf** file that contains data about the BRM components running on the system.

To run the **pin_ctl** utility, set the PERL5LIB environment variable to point to the third-party application's install directory. To do so, perform one of the following:

- Add the following paths to the PERL5LIB environment variable for the root account on each managed node:
  - *BRM_home***/ThirdPartyApps/tools/PerlLib**
  - *BRM_home***/bin**
- Before you deploy the **call_pin_ctl** script in *BRM_SPI_install_directory***/bin**, add the following paths to the PERL5LIB variable in the script:
  - *BRM_home***/ThirdPartyApps/tools/PerlLib**
  - *BRM_home***/bin**

You can configure the **pin_ctrl** utility as follows:

- Customizing the List of Components to Start or Stop
- Starting and Stopping Optional Components by Using pin_ctl
- Creating a Custom Component List
- Customizing the Components List

- Customizing the pin_ctl Startup Configuration
- Customizing the pin_ctl Utility Environment Variables
- Configuring the Start and Stop Validation Settings for pin_ctl
- Using Custom pin_ctl Configuration Files

Customizing the List of Components to Start or Stop

You can customize the components that are included in the **pin_ctl** utility **all** component.

1. Open the **pin_ctl.conf** file in *BRM_home*/**bin**.
2. Find the following lines in the file:

```
# List of services to be part of all [Optional].
# Mention the service names separated by a space.
# '=' should be used to create an alias for 'all'.
# For example, all=my_all
# all=my_all dm_oracle dm_email cm cmmp formatter

all dm_oracle dm_email cm cmmp formatter
```

3. After **all**, enter each component that you want to start with the **all** command:

```
all component1 component2 component3 ...
```

   where *componentX* is the component you want to add. For a list of valid component values, see "pin_ctl".

> **Note:**
>
> Make sure the components are in the order in which you want them started. The order is reversed when the components are stopped.

4. Save and close the file.

Starting and Stopping Optional Components by Using pin_ctl

The default **pin_ctl.conf** file is configured to start BRM system components only. To configure **pin_ctl.conf** to start an optional component, such as the Synchronization Queue DM (**dm_aq**):

1. Open the **pin_ctl.conf** file in *BRM_home*/**bin**.
2. Add the following line to the components list:

```
start_sequence service_name [=alias_name|:java|:app|->dependent_service]
```

   where:

   - *start_sequence* is the start and stop sequence number. This determines the order in which components are started or stopped.
   - *service_name* is the name of the optional component.
   - **=***alias name* indicates that *service_name* is different from the standard service name. For example:

cm_1=cm

cm_2=cm

where **cm_1** and **cm_2** are **cm** services.

- **:java** indicates that the component is Java-based.
- **:app** indicates that the component executable is located in the *BRM_home***/apps** directory.
- **->***dependent_service* specifies one or more components that *service_name* is dependent on. This indicates that *dependent_service* must start before *service_name* is started.

For example, to add **dm_aq** to the components list:

```
4 dm_aq
```

3. Add the following line to the startup configuration section of the file:

```
start_component cpidproc:searchpattern:pidvarname cport:port_number
[testnap:directory_name]
```

where:

- *start_component* is the name of the start command for the optional component, such as **start_dm_aq**. It must be unique; if not, the last parsed definition is used.
- **cpidproc:***searchpattern* is a simple process name matching filter.
- *pidvarname* is a partial match for the **pidfile** variable from ${*program_name*}. If you enter nothing (which is recommended), the default is **PID$**, which matches **CMPID** in **$PIN_LOG/cm/cm.pid**.
- **cport:***port_number* is the component port number.
- **testnap:***directory_name* runs the **testnap** utility in the specified directory. The directory is relative to *BRM_home***/sys**.

For example, to enter a startup configuration for **dm_aq**:

```
start_dm_aq cpidproc:dm_aq: cport:--DM_AQ_PORT__
```

4. Save and close the file.

Creating a Custom Component List

You can create aliases for custom lists of components that are controlled by the **pin_ctl** utility **all** component. For example, if you define an alias named **my_all**, you can start a custom group of components by running the following command:

```
pin_ctl start my_all
```

1. Open the **pin_ctl.conf** file in *BRM_home***/bin**.
2. Find the following lines in the file:

```
# List of services to be part of all [Optional].
#       Mention the service names separated by a space.
# '='   should be used to create an alias for 'all'.
#       For example, all=my_all
# all=my_all dm_oracle dm_email cm cmmp formatter

all dm_oracle dm_email cm cmmp formatter
```

3. Add the following line at the end of the section:

   ```
   all=alias component1 component2 ...
   ```

   where:

   - *alias* specifies the name of your customized **all** command. For example, **my_all**.
   - *componentX* is the component you want to add. For a list of valid component values, see "pin_ctl".

   > **Note:**
   >
   > Make sure the components are in the order in which you want them started. The order is reversed when the components are stopped by using the custom **all** command. Use a space to separate component names.

4. Save and close the file.

Customizing the Components List

The components list in the **pin_ctl.conf** file lists the BRM system components. For example:

```
1 dm_oracle
1 dm_email
1 dm_fusa
1 dm_invoice
...
```

If you have a high-availability system that includes duplicate instances of components, you can edit the **pin_ctl.conf** file to customize the components list. For example:

```
1 dmo1=dm_oracle
1 dmo2=dm_oracle
1 dm_eai_1=dm_eai
1 dm_eai_2=dm_eai
2 cm_1=cm->dm_oracle
2 cm_2=cm->dm_oracle
3 cm_proxy_1=cm_proxy
3 cm_proxy_2=cm_proxy
3 cmmp_1=cmmp
3 cmmp_2=cmmp
```

To customize the component list:

1. Open the **pin_ctl.conf** file in *BRM_home*/**bin**.
2. Find the following lines in the file:

   ```
   # The format of entry for each service is ,
   # start_sequence service_name [=<alias_name>|:java|:app|-><list of services
   depends on>]
   # The start sequence is a mandatory field, which gives sequence to start/
   stop [Mandatory].
   # Sequence is a numerical value, and starts from 1. The service should be
   specified
   ```

```
# in the assending order based on the sequence number.
# Mention the service name. This service_name is mandatory field [Mandatory].
# NOTE: Start sequence and Service name should be separated by a space.
# '='   should be used if service name is different with standard service names
[Optional].
# For example, cm2=cm
# Here, cm2 is the service which is of cm category.
# This is useful when multiple CMs/DMs are installed.
# :app  should be used if its located in BRM_home/apps directory [Optional].
# :java should be used if its a java based service [optional].
# ->    should be used if the current service has any dependencies [Optional].
# This is generally useful in WINDOWS.
```

3. Add the following line for each component in your system:

   start_sequence service_name [**=***alias_name*|**:java**|**:app**|**->***dependent_service*]

   where:

   - *start_sequence* is the start/stop sequence number.
   - *service_name* is the component name.
   - **=***alias name* indicates that *service_name* is different from the standard service name. For example:

     cm_1=cm

     cm_2=cm

     where **cm_1** and **cm_2** are **cm** services.
   - **:java** indicates that the component is Java-based.
   - **:app** indicates that the component executable is located in the *BRM_home***/apps** directory.
   - **->***dependent_service* specifies one or more components that *service_name* is dependent on. This indicates that *dependent_service* must start before *service_name* is started.

4. Save and close the file.

Customizing the pin_ctl Startup Configuration

The **pin_ctl.conf** file includes startup configurations for system components. For example:

```
start_cm          cpidproc:cm:         cport:2224     testnap:test
```

These configurations are created automatically during installation, but you can change them. For example, if you use a high-availability system with duplicate processes, you should change the component names. In the following example, the Oracle DM name in the component list is **dmo1**, so the startup configuration has been changed to match:

```
start_dmo1   cpidproc:dmo1:  cport:12432
```

1. Open the **pin_ctl.conf** file in *BRM_home***/bin**.
2. Edit the file.

   The syntax is as follows:

   start_component **cpidproc:***searchpattern*:*pidvarname* **cport:***port_number* [**testnap:***directory_name*]

where:

- *start_component* is the name of the start command. It must be unique; if not, the last parsed definition is used.
- **cpidproc:***searchpattern* is a simple process name matching filter.
- *pidvarname* is a partial match for the **pidfile** variable from ${*program_name*}. If you enter nothing (which is recommended), the default is **PID$**, which matches **CMPID** in **$PIN_LOG/cm/cm.pid**.
- **cport:***port_number* is the component port number. This value is entered automatically during installation.
- **testnap:***directory_name* runs the **testnap** utility in the specified directory. The directory is relative to *BRM_home***/sys**.

3. Save and close the file.

Customizing the pin_ctl Utility Environment Variables

Some BRM components need environment variables set before starting. You can edit the **pin_ctl.conf** file to change the environment variables if yours are different from the default settings.

1. Open the **pin_ctl.conf** file in *BRM_home***/bin**.
2. To define environment variables for BRM components, find the following lines in the file:

```
# List of all environment variables which needs to be set
# or override during the execution a particular process
# The syntax for setting or overriding the environment variable will be,
# program_name env_platform:OS env_variable:ENV_VAR    env_val:ENV_VAL

#common env_platform:solaris env_variable:EXAMPLE env_val:example
```

3. Add the following line for each BRM component that requires an environment variable:

```
component env_platform:operating_system env_variable:environment_variable
env_val:value
```

where:

- *component* is the BRM component that uses the environment variable (for example, **cm**). Use **common** to apply the environment variable to the entire system.

  For a list of component values, see "Parameters for Components".
- *operating_system* can be **linux**, **solaris**, or **common**.
- *environment_variable* specifies the name of the environment variable to set before starting *component*.
- *value* specifies the environment variable value.

For example, the following line sets the NLS_LANG environment variable before starting any BRM component:

```
common env_platform:common env_variable:NLS_LANG
env_val:AMERICAN_AMERICA.AL32UTF8
```

4. Save and close the file.

Configuring the Start and Stop Validation Settings for pin_ctl

You can configure the validations **pin_ctl** performs when starting and stopping components, including the following:

- How long the utility waits before checking whether an action is complete.
- The maximum number of times the utility checks whether an action is complete.
- The home directory for the specified component. This overrides the *BRM_home* value.
- The home log directory for the specified component. This overrides the $PIN_LOG value for the specified component.

To specify the validation settings used when **pin_ctl** starts and stops components:

1. Open the **pin_ctl.conf** file in *BRM_home***/bin**.
2. Find the following lines in the file:

```
# This sections will be used to have different settings for each service like
# 1. waittime -- number of seconds to be waited
# 2. iterations -- Number of times to be checked
# 3. pin_home_dir -- BRM_home path
# 4. pin_log_dir -- BRM_home/var/component path
# All these are optional, if these are not set then default values will be used.
```

3. Add the following line for each component that you want to override the default values:

```
settings component waittime:wait iterations:value pin_home_dir:path
pin_log_dir:log_file_path
```

   where:

   - *component* is the BRM component. For a list of valid values, see "Parameters for Components".
   - *wait* is the number of seconds to wait before checking whether an action is complete. The default is **5**.
   - *value* is the maximum number of times to check whether an action is complete. The default is **5**.
   - *path* is the home directory. This overrides the *BRM_home* value.
   - *logpath* is the BRM log file home. The default is the value set in the $PIN_LOG environment variable. You must change this only if you use a different directory than the default directory.

   For example:

```
settings dm_oracle waittime:5 iterations:5 pin_home_dir:BRM_home
pin_log_dir:$PIN_LOG
```

4. Save and close the file.

Using Custom pin_ctl Configuration Files

You can create custom **pin_ctl** configuration files to run different configurations of the same system.

1. Create a custom configuration file in *BRM_home***/bin**. You can copy and rename the **pin_ctl.conf** file.

2. Use the **-c** *file_name* parameter when you run the **pin_ctl** utility. For example:

```
pin_ctl cstart all -c pin_ctl_batch.conf
```

Starting BRM Components Automatically

You can configure BRM components to start automatically when you restart a system by adding component start scripts to the operating system startup scripts, such as the **/etc/rc2** script. You can also start components automatically from **cron** jobs.

> **Note:**
>
> When you set up the BRM system to start automatically, ensure the database starts *before* any Data Manager that connects to the database. If the DM starts first, it reports an error when it cannot find the database. For more information on component start sequences, see "Order of Starting and Stopping Components".

To add a component to the startup script:

1. As user **root**, run the installation script for the component (for example, **install_dm_fusa** or **install_cm**).

   These scripts are in *BRM_home***/bin**.

2. (Optional) To avoid problems with file security and permissions, set all component processes to start as user **pin** (or another name you choose) rather than as user **root**; add the following line to the initialization file for the operating system, before the lines that start the components:

```
su - pin
```

Starting Multiple Families of BRM Components on the Same Computer

Each BRM component can be part of a family of components. Each family includes a parent process and one or more child processes. When you start the parent process, BRM starts child processes automatically.

You can run multiple families of a BRM process on the same computer. For example, you can start another instance of the Paymentech DM on a computer that is already running the Paymentech DM.

To run multiple families, put each family in a separate directory with its own configuration file (**pin.conf** or **Infranet.properties**). Each family's configuration file must point to a unique port to avoid conflicts when you start the processes. Each configuration file should point to a unique **pinlog** file as well.

For information on configuring the port and on the location of the log file for a process, see the explanatory text in the configuration file for that process.

Confirming That a BRM Component Started Successfully

To verify that a component started successfully, perform one or more of these checks:

- For supported components, use the **pin_ctl** utility with the **status** action. See "pin_ctl".
- Look for the startup timestamp in the **.log** file. For more information on BRM log files, see "Using Logs to Monitor Components".
- Confirm that the **pid** file for the component contains the process ID (PID).

  pid files are generated in *BRM_home***/var/***component* (for example, *BRM_home***/var/dm_oracle**).
- Use the **ps** command to check the component process status.
- **(Solaris and Linux)** You can confirm that a shared memory segment has been allocated for the component process by using the **ipcs** command.

> **Note:**
>
> The **ipcs** command does not show the shared memory segment unless you run it as **root** or **pin** or you use the -**m** parameter.

Stopping a Process by Using Commands

In addition to using the **pin_ctl** utility **halt** command, you can stop a component with a direct command. Use **ps** to get the process ID (PID) of the process, and then use the **kill** command to stop the process.

> **Note:**
>
> Stopping the CM parent also stops the CM children. If you kill a child CM, a new opcode call starts a new child CM. This is because the parent CM is still active, and can automatically start a child CM when you run an opcode.

In rare cases, you might be left with an allocated but unused shared memory block. Use the **ipcs** command to detect an allocated block; use the **ipcrm** command to remove it.

2
Monitoring Your BRM System

Learn how to monitor your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

See also "Using Logs to Monitor Components", "Monitoring Connection Manager Activity", and "Monitoring Data Manager Activity".

Monitoring Hardware and Operating Systems

To monitor your system using standard tools, use monitoring utilities such as **vmstat**, **sar**, and **top** on UNIX.

On Solaris systems, use **sysdef** to find information about kernel parameter settings. This is especially useful for determining if per-process shared memory, file descriptor, or thread limits are adequate. **pmap** is useful for separating memory usage into total, resident, shared, and private.

Checking the Number and ID of a BRM Process

You can check the number of processes running for the CM or a DM. The number should match the number specified in the configuration file (**pin.conf**) for that component. If not, the processes either did not start or have stopped. You can also look at the process ID (PID) for each process.

Enter the following command:

```
ps -ef | grep process
```

The system shows each process and its ID.

For example, to show the processes running for the Paymentech DM, enter the following command:

```
ps -ef | grep dm_fusa
```

Checking the Version Numbers of Components

You can check the version numbers of all BRM components installed on a machine by using the **viewInventory.sh** script.

The following information for each component and patch installed on a machine are available in the *Oracle_home*/**inventory/registry.xml**, where *Oracle_home* is the directory in which the Oracle products are installed:

- Distribution name
- Feature-sets name
- Feature status
- Version number
- Installation time

  For example:

```
<feature status="installed" name="BRMServer" version="12.0.0.1.0">
   <sessions>
      <session id="1" date="2018-09-08T18:53:47.492-07:00" action="install"/>
   </sessions>
</feature>
```

> **Note:**
>
> Use this file or script whenever you are working with Oracle Technical Support to help them re-create your system environment and troubleshoot your problems.

Checking BRM Component Version Numbers

To check which BRM components are installed on your machine, go to the *Oracle_home*/**oui/bin** directory and run the **viewInventory.sh** script:

```
sh viewInventory.sh
```

BRM displays the versions of all products installed on your machine.

> **Note:**
>
> The **viewInventory.sh** script does not display information about the uninstallation of any BRM component. Only installation information is displayed.

Dumping Business Parameters in XML Format

To dump BRM business parameters (**/config/business_params** objects) in XML format, use the **pin_cfg_bpdump** utility. For more information about business parameters, see "business_params Reference".

You can use the output as input to another application or utility, such as a diagnostic application. You can also direct the XML output to a file. For example, to direct the output to a file named **myfile.xml** in the same directory in which the utility is run, enter the following command:

```
pin_cfg_bpdump > myfile.xml
```

For each **/config/business_params** object, the utility outputs a **<RESULTS>** element that supplies identifying information about the object. The **<RESULTS>** elements include a **<PARAMS>** element for each parameter they include. A **<PARAMS>** element provides the parameter description, name, type, and value.

To dump business parameters by using the **pin_cfg_bpdump** utility:

1. Go to the *BRM_home*/**diagnostics/pin_cfg_bpdump** directory.
2. Run the following command:

   ```
   pin_cfg_bpdump
   ```

   To direct the output to a file, use the following syntax:

   ```
   pin_cfg_bpdump > file_name
   ```

   where *file_name* is the name of a file in the same directory in which the utility is run.

Monitoring Global Transactions

The DBA_2PC_PENDING view stores detailed information about each **Pending** global transaction.

DBA_2PC_PENDING is a static data dictionary view in the BRM database. To enable Oracle DM processes to access this view, your system administrator must grant read privileges to the BRM database user.

Table 2-1 describes the columns in the DBA_2PC_PENDING view.

Table 2-1 DBA_2PC_PENDING View Description

Column Name	Description
LOCAL_TRAN_ID	For extended architecture (XA) transactions, specifies the branch qualifier element of an XA transaction ID (XID), which uniquely identifies the local BRM database branch of the XA transaction.
GLOBAL_TRAN_ID	For XA transactions, specifies the global transaction ID element of an XID, which uniquely identifies the XA transaction.
STATE	Specifies the transaction's state, which can be one of the following values: • **Collecting:** Applies only to the global coordinator or local coordinators. The node is currently collecting information from other database servers before it can decide whether it can prepare. • **Prepared:** The node has prepared and may or may not have acknowledged this to its local coordinator or transaction manager with a prepared message. However, no commit request has been received. The node remains prepared, holding any local resource locks necessary for the transaction to commit. • **Committed:** The node (any type) has committed the transaction, but other nodes involved in the transaction may not have done the same. That is, the transaction is still pending at one or more nodes. • **Forced commit:** A pending transaction can be forced to commit at the discretion of a database administrator. This entry occurs if a transaction is manually committed at a local node. • **Forced rollback:** A pending transaction can be forced to roll back at the discretion of a database administrator. This entry occurs if this transaction is manually rolled back at a local node.

Table 2-1 (Cont.) DBA_2PC_PENDING View Description

Column Name	Description
MIXED	YES means that part of the transaction was committed on one node and rolled back on another node.
TRAN_COMMIT	Specifies the transaction comment or, if transaction naming is used, the transaction name.
HOST	Specifies the host machine name.
COMMIT#	Specifies the global commit number for committed transactions.

3

Tracking Failed BRM Operations

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) to collect details about failed operations and store them in the database.

Topics in this document:

About Tracking Failed BRM Operations

BRM operations may occasionally fail to process completely. For example, a payment could fail due to an insufficient balance or an incorrect account address, or a loan request could fail because it doesn't meet the eligibility requirements. Likewise, a network connection could drop while processing a transaction. You can configure BRM to store information about these failed operations in the BRM database, so you can view them for analysis and reporting, or reprocess them at a later time.

BRM can record details about failed operations that occur in the following components:

- Billing Care REST API

  To configure the Billing Care REST API to record details about failed REST requests, see "Recording Billing Care REST API Request Failures" in *Billing Care SDK Guide*.

- BRM multithreaded application (MTA) utilities

  To configure MTA utilities to record failed operations, see "Configuring MTA Utilities to Record Operation Failures".

- Custom client applications

  To configure a custom client application to record and retrieve failed operations, use the PCM_OP_ACT_REQUEST_CREATE and PCM_OP_ACT_REQUEST_RETRIEVE opcodes. See "Managing Operation Failure Records" in *BRM Opcode Guide*.

When configured to do so, BRM stores details about failed operations, such as the request payload and the error that occurred, in **/request/failed**, **/request/failed/rest**, and **/request/failed/opcode** objects. In multischema systems, the **/request/*** object is stored in the schema in which the failed operation occurred. For more information about these objects, see *Storable Class Reference*.

The Billing Care REST API and MTA utilities remove sensitive information, such as passwords, from request payloads before they are stored in the database. If you use a custom client application, you must configure it to mask sensitive information before calling the PCM_OP_ACT_REQUEST_CREATE opcode. See "Recording Failed Operations" in *BRM Opcode Guide*.

Configuring MTA Utilities to Record Operation Failures

To configure BRM MTA utilities to record details about failed operations and transactions, include the **-record_failure true** parameter in the utility's command line. For example, to record details about failures that occur when generating bills for active accounts, you would run the following:

```
pin_bill_accts -active -record_failure true
```

Table 3-1 lists the MTA utilities that support the **-record_failure** parameter.

Table 3-1 Utilities Supporting Operation Failure Records

Utility Type	Utility Name
Accounts Receivable	**pin_apply_bulk_adjustment** **pin_balance_transfer** **pin_monitor_balance** **pin_roll_up_ar_items**
Billing	**pin_bill_accts** **pin_cycle_fees** **pin_cycle_forward** **pin_rollover** **pin_roll_up_ar_items** **pin_update_journal**
Collections Management	**pin_collections_process** **pin_collections_send_dunning**
Customer Management	**pin_contracts** **pin_deferred_act** **pin_state_change**
Invoicing	**pin_inv_accts** **pin_make_corrective_bills** **pin_upd_assoc_bus_profile**
Payment	**pin_cc_migrate** **pin_collect** **pin_deposit** **pin_deposit_calc_interest** **pin_deposit_release_purchased_deposit** **pin_deposit_transfer_deposit** **pin_installment_status_change** **pin_installments** **pin_refund**
Analytics	**pin_generate_analytics**

4
Backing Up and Restoring Your BRM System

Learn how to back up and restore your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

About Backing Up and Restoring BRM

To prevent any data loss and minimize the impact of software or hardware failure, back up your system immediately after installing or updating the system.

Create a backup of the BRM configuration files and BRM installation area (the BRM installation directory and its content: *BRM_home*). You can perform this backup as soon as you install and configure BRM. In particular, make sure you back up all customized files, including source code, policy, **start_all**, **pin.conf**, **pin_ctl.conf**, **pin_setup.values**, and **Infranet.properties** files.

Repeat the backup process whenever you customize or update the BRM configuration files. For instructions on backing up the standard configuration, see "Backing Up BRM Configuration".

> **Note:**
>
> Store this backup in a safe location. The data in these files are necessary if you encounter any operational or system issues.

After you complete the backup, you can use the backup configuration directory and installation area to restore your system when needed. For instructions on restoring the standard configuration, see "Restoring BRM Configuration".

If you do not back up your BRM system regularly, you need to reinstall and reconfigure the system if it is corrupted due to operational or system errors. Reinstalling and reconfiguring eliminates any chance of recovering and reprocessing data processed by the BRM system at the time of the error.

If you require BRM technical support, email a copy of your backup configuration directory to your Oracle Global Support representative. If you want to send a copy of your configuration directory, use the **tar** command to create an archived version of that directory.

If you want to make a complete offline backup of your BRM or Pipeline Manager database, use the appropriate backup tools for your database version and ensure that the backup is completely valid and usable. The backup must contain both the database definition and all

the database contents. For more information, see the discussion about performing full database backups in your database software documentation.

Backing Up BRM Configuration

To create a backup of the BRM configuration, see the following:

- Backing Up BRM Files
- Backing Up Pipeline Manager Files
- Backing Up Rated Event Loader Files

Backing Up BRM Files

To back up BRM configuration:

> **Note:**
>
> In multischema systems, perform this task first on the primary BRM installation machine and then on the secondary BRM installation machines.

1. On the machine on which you installed BRM, copy the **vpd.properties** file by running the following command:

   ```
   cp vpd.properties vpd.properties_75
   ```

2. Go to the *BRM_home* directory.
3. Copy the content of the *BRM_home* directory to a new directory by running the following command:

   ```
   cp -R BRM_home NewName
   ```

   where *NewName* is the name for the new directory.

   > **Note:**
   >
   > You can remove the BRM log files from the backup directory.

4. Create an archive of the entire directory and the **vpd.properties** file by running the following command:

   ```
   tar cvf NewName.tar.gz NewName
   tar cvf vpd.properties_75.tar.gz vpd.properties_75
   ```

5. Store the backup copies in a location outside of the BRM system by running the following command:

   ```
   mv NewName.tar.gz New_Directory
   mv vpd.properties_75.tar.gz New_Directory
   ```

   where *New_Directory* is the new location that is outside of the BRM system.

A compressed TAR file, of all copied files, is created with the extension **tar** in the new location specified (for example, **brm_backup/brm_home.tar.gz**).

Backing Up Pipeline Manager Files

If you are using Pipeline Manager for usage charging, back up the Pipeline Manager files.

> **Note:**
>
> In multischema systems, perform this task first on the primary BRM installation machine and then on the secondary BRM installation machines.

To back up Pipeline Manager files:

1. On the machine on which you installed Pipeline Manager, copy the **vpd.properties** file by running the following command:

   ```
   cp vpd.properties vpd.properties_75
   ```

2. Copy the content of the *Pipeline_home***/ifw** directory to a new directory by running the following command, where *Pipeline_home* is the directory in which Pipeline Manager is installed:

   ```
   mv Pipeline_home/ifw Pipeline_home/ifw_75
   ```

   > **Note:**
   >
   > You can remove the log files from the backup directory.

3. Create an archive of the entire directory and the **vpd.properties** file by running the following command:

   ```
   tar cvf ifw_75.tar.gz ifw_75
   tar cvf vpd.properties_75.tar.gz vpd.properties_75
   ```

4. Store the backup copies in a location outside of the BRM system by running the following command:

   ```
   mv Pipeline_home/ifw_75.tar.gz New_Directory
   mv vpd.properties_75.tar.gz New_Directory
   ```

   A compressed TAR file, of all copied files, is created with the extension **tar** in the new location specified (for example, **pipeline_backup/ifw_75.tar.gz**).

Backing Up Rated Event Loader Files

By default, Rated Event (RE) Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down abnormally.

To prevent data loss when your system shuts down, archive all the successfully loaded files.

To back up the RE Loader files:

1. Create an archive of the entire directory in which the RE Loader files are stored by running the following command:

   ```
   tar cvf RE_Loader_Dir.tar.gz NewName
   ```

2. Store the backup copy in a location outside of the BRM system by running the following command:

   ```
   mv NewName.tar.gz New_Directory
   ```

   A compressed TAR file, of all copied files, is created with the extension **tar** in the new location specified (for example, **RE_Loader_backup.tar.gz**).

Restoring BRM Configuration

To restore the BRM configuration:

1. Ensure that no users are logged in.

   Users include customers, client applications, customer service representatives (CSRs), and so on.

2. Stop all BRM processes.

   Only the database instances should be running during the patch installation.

   For instructions, see "Starting and Stopping the BRM System".

3. Delete or rename the damaged *BRM_home* directory:

   - To delete, run the following command:

     ```
     rm -R BRM_home
     ```

   - To rename, run the following command:

     ```
     mv BRM_home New_Name
     ```

4. Retrieve the backup tar file of the *BRM_home* directory.

5. Extract from the TAR file the backup copy of the directory by running the following command:

   ```
   tar xvf BRM_home.tar.gz
   ```

   The command recreates the copy of the *BRM_home* directory.

6. Repeat steps 3 to 5 with *Pipeline_home* to restore the Pipeline Manager configuration.

7. Repeat steps 3 to 5 with **vpd.properties_75** to restore the **vpd.properties** file for BRM and Pipeline Manager.

   > **Note:**
   >
   > If you have BRM and Pipeline Manager installed on separate machines, perform this step on both the machines.

8. Repeat steps 3 to 5 with *RE_Loader_Dir* to restore the RE Loader files.

9. Start BRM processes and Pipeline Manager.

For instructions, see "Starting and Stopping the BRM System".

All updates, which had been temporarily disrupted due to the shutdown, are processed upon restart.

5

Using Logs to Monitor Components

Learn how to use log files to monitor your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Using Logs to Monitor Components

BRM records system activity in log files. One log file is generated for each component or application. Review these files daily to monitor your system and detect and diagnose system problems. You can also:

- Write scripts to look for certain conditions, such as types or numbers of errors, and to notify you when these conditions occur.
- Record opcode calls in the CM log file. See "Recording Opcode Calls in the CM Log File".

BRM generates log files for system components, applications, and client applications.

For system processes (or threads) such as CMs and DMs, BRM uses two types of log files:

- Those that record normal startup activity are named *program*.**log** (for example, **cm.log**, **js.log**, and **dm.log**).
- Those that record activity, such as error conditions, while the system is running. These pinlogs are named *program***.pinlog** (for example, **cm.pinlog**, **js.pinlog**, and **dm_oracle.pinlog**).

For BRM applications, log files are named *program***.pinlog** (for example, **pin_billd.pinlog**). If an application is missing a configuration file (**pin.conf**) or if the application fails before it can read the configuration file, it records errors in the **default.pinlog** log file.

> **Note:**
>
> Calls made by opcodes to get data from objects are not recorded in log files.

BRM Java-based applications, such as Customer Center and Configuration Center, by default do not use log files. However, you can enable error logging by adding entries to the **Infranet.properties** file that provide configuration information when the application starts.

Log files for system components are stored in *BRM_home***/sys/***component*. For example, the CM log file is in *BRM_home***/sys/cm**.

If there is no log file in *BRM_home***/var/***component*, the **default.pinlog** file is used instead. It is stored in *BRM_home***/sys/***component*. For example, the CM **pinlog** file is *BRM_home***/sys/cm/default.pinlog**.

For an application or client application log file, the default location is the directory from which the program was started.

You can leave log files in their default locations or move them.

Large log files degrade system performance. Check the sizes of log files periodically and delete or archive large files. When you delete or rename a log file, a new empty file is created as soon as a new log entry is created *and* either a maximum of four hours have elapsed or the application is stopped and restarted. Be especially vigilant when using new custom applications, which commonly makes log files grow quickly.

Log files should be archived weekly to a safe storage area.

Changing the Name or Location of a Log File

To change the name or location of the **pinlog** file for a component or application:

1. Open the configuration file (**pin.conf** or **Infranet.properties**) for the component or application.
2. Change the relevant entry:
   - **logfile**: Applications
   - **cm_logfile**: CM
   - **dm_logfile**: DM
3. Enter the desired name and directory for the log file.
4. Save and close the file.
5. Stop and restart the component or application.

> **Note:**
>
> - You can change the name of the default application's log file by using the PIN_ERR_SET_LOGFILE function. See *BRM Developer's Reference*.
> - For Payment Tool, you cannot change the name of the log file. For Java-based BRM client applications, use an **Infranet.properties** file to specify the name and location of a log file.

Setting the Reporting Level for Logging Messages

By default, BRM components report error messages, and BRM applications report both error and warning messages. You can set BRM to report debugging messages or to not report errors. The four levels of error reporting are:

- **0** = no logging.
- **1** = (default) log error messages only.
- **2** = log error messages and warnings.
- **3** = log error, warning, and debugging messages.

> **Note:**
>
> To avoid performance degradation, use only level 3 logging for debugging.

A common implementation is to use 2 and use a script to detect and act on warning messages.

To change the severity level for logging:

1. Open the configuration file (**pin.conf** or **.properties**) for the component or application.
2. Edit the **loglevel** entry. The notes in the configuration file define the options.
3. Save and close the file.
4. Stop and restart the component or application.

Setting the Log Level for a Specific Opcode

You can record debug-level information for a specified opcode without having to reset the default system log level. This enables you to monitor the activity of a specific opcode (and any opcode it calls) without impacting system performance.

When you enable opcode logging, the logging level is increased to debug level 3 for the specified opcode only; all other opcodes are logged at the level specified in the CM **pin.conf** file.

You can define how many times during a CM session the debug-level reporting occurs for the specified opcode before the default reporting level is restored. This enables you to increase the logging level without having to stop and restart the CM to reset it to the default level.

1. Open the CM **pin.conf** file in *BRM_home*/**sys/cm**.
2. Set the **pinlog_debug_opcode** entry:

   ```
   cm pinlog_debug_opcode opcode
   ```

   where *opcode* is the opcode name or opcode number.

> **Note:**
>
> If this entry is not set, BRM uses the **loglevel** entry in the CM **pin.conf** file to determine the log level.

3. Set the **pinlog_debug_op_count** entry:

   ```
   cm pinlog_debug_op_count number
   ```

   where *number* is the number of times the opcode is recorded at the debug level before the default log level is restored.

4. Save and close the file.
5. Restart the CM.

Recording Opcode Calls in the CM Log File

You use the **enable_pcm_op_call_stack** and **max_pcm_op_call_stack_entries** CM **pin.conf** entries to record opcodes in the CM log file.

When **enable_pcm_op_call_stack** is enabled, the opcodes that are called by BRM clients are recorded in the CM log file.

See "Connection Manager (CM) pin.conf Entries".

About Formatting Log Files

You can format a log file to improve readability and traceability of errors by using the **splitPinlog** script. This script splits a log file into multiple files, one for each combination of process ID (PID) and thread ID (TID) based on the information in the header of the **pinlog** entries.

To format a log file:

1. Go to the *BRM_home***/bin** directory.
2. Run the following Perl script:

   ```
   splitPinlog original_pinlog_file
   ```

The Perl script creates a file with the name *original_pinlog_file*.*pid*.*tid*.**pinlog**.

For example, running the command:

```
splitPinlog cm.pinlog
```

results in these file names:

- **cm.pinlog.342353.12.pinlog**
- **cm.pinlog.342353.13.pinlog**

6
Monitoring Connection Manager Activity

Learn how to monitor Oracle Communications Billing and Revenue Management (BRM) Connection Managers (CMs).

Topics in this document:

Manually Checking the Status of the CM

You can monitor the operation of the CM by checking the status at regular intervals and comparing the results with what you expect.

To check the status of the CM:

1. Find the process ID (PID) of the main CM process by looking in the **pid** file for the CM in *BRM_home*/**sys/cm**.
2. Enter the following command:

   ```
   kill -USR1 PID_of_CM
   ```

BRM displays a report on the CM, which shows information about the master CM such as the version and the number of children. If there are CM children, the rest of the reports consist of a single line for each child showing the state, the IP address and port for the application, and the IP address and port of the current DM connection.

Table 6-1 describes the state values:

Table 6-1 CM State Values

Value	Description
1	Reading from (or waiting to read from) the application
2	Starting to process the operation
3	Facilities Module processing in progress (if going to FM)
4	Facilities Module processing done, sending response
5	Finding DM address (if going to DM)
6	Sending operation to DM
7	Waiting on DM
8	Forwarding DM response to application
9	Cleaning up after the operation

Table 6-1 (Cont.) CM State Values

Value	Description
10	Shutting down the child CM
11	Starting the child CM

Enabling Java PCM Clients to Use Operating System TCP/IP Keepalive Parameters

If a connection for a Java PCM client is not in use for some time, a **BAD_READ** error may result. If this becomes a recurring problem, you can enable the client to use the underlying operating system TCP/IP keepalive parameters such as keepalive time, keepalive interval, and keepalive retry.

To enable Java PCM clients to use operating system TCP/IP keepalive parameters:

1. Open the **Infranet.properties** file of the Java PCM client.

   A Java PCM client is any Java client application that communicates with BRM by using the Java Portal Communication Module (Java PCM) API (for example, Developer Center, or a custom application).

2. Add the following entry:

   ```
   infranet.pcp.socket.keepalive.enabled=true
   ```

   - **true** enables Java PCM clients to use operating system TCP/IP keepalive parameters.
   - **false** prevents Java PCM clients from using operating system TCP/IP keepalive parameters.

   By default, BRM prevents Java PCM clients from using operating system TCP/IP keepalive parameters.

3. Save and close the file.

Setting the CM Log Time Resolution

By default, the time resolution in CM log files is in seconds. If you need a higher resolution to help diagnose performance issues, change the resolution to milliseconds.

To set the CM log time resolution:

1. Open the CM **pin.conf** file in *BRM_home***/sys/cm**.
2. Change the value of the **cm_logformat** entry from **0** to **1**, where **0** sets the log time resolution to seconds and **1** sets the log time resolution to milliseconds.
3. Save and close the file.
4. Stop and restart the CM.

Setting a Timeout Value for Requests Sent to the CM

BRM client applications process requests in a synchronous mode; that is, they wait for a response from the CM before sending the next request. Therefore, if there is an error on the server side, the client application has to wait indefinitely. To prevent this problem, you can set a timeout value for requests sent to the CM. If the CM does not respond within the time specified, the PCP connection layer returns an error message to the client application and closes the connection.

To specify a timeout value, configure your client applications as follows:

- For BRM client applications that use a configuration (**pin.conf**) file:
  1. Open the **pin.conf** file in a text editor.
     By default, the **pin.conf** file is in *BRM_home***/apps/***application_name*, where *application_name* is the name of the application, such as **pin_billd**.
  2. Add the following entry to the file:

     ```
     - nap pcm_timeout_in_msecs milliseconds
     ```

     where *milliseconds* is the number of milliseconds to wait before returning an error message and closing the connection.
  3. Save and close the file.
- For Java Connector Architecture (JCA) Resource Adapter:
  1. Open the Adapter deployment descriptor (**ra.xml**) file in a text editor.

     > **Note:**
     >
     > By default, the **ra.xml** and **weblogic-ra.xml** files are packaged along with the adapter archive (RAR) file.

  2. Set the following parameter in the file:

     ```
     PcmTimeout milliseconds
     ```

     where *milliseconds* is the number of milliseconds to wait before returning an error message and closing the connection.

     The default value is 0, which means JCA waits infinitely for the response from CM.
  3. Save and close the file.
  4. Open the Oracle WebLogic Server deployment descriptor (**weblogic-ra.xml**) file in a text editor.
  5. Repeat steps 2 and 3.

> **Note:**
>
> The timeout value specified in the files is used for all open connections. If a timeout value is set for a connection in the application itself, that value overrides the value in the files.

For information on setting timeout values for each connection in your custom C and Java client applications, see "Implementing Timeout for Requests in Your Application" and "Specifying a Timeout Value for Requests" in *BRM Developer's Guide*.

Configuring Multilevel CM Timeout for Client Requests

You can configure the CM to use two timeouts for handling client requests:

- A short (suspect) timeout
- A long (failover) timeout

A Short (Suspect) Timeout

When this timeout period expires, the request for the DM connection is placed in a suspect state, the current transaction is stopped, and the request is returned to the client with the PIN_ERR_TIMEOUT and PIN_ERRCLASS_SYSTEM_SUSPECT errors.

To configure the suspect timeout:

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

   ```
   pcm_suspect_timeout_in_msecs milliseconds
   ```

   where *milliseconds* is the number of milliseconds in the suspect timeout period.

   > **Note:**
   >
   > The value of this entry must be smaller than the value of the **pcm_timeout_in_msecs** entry.

3. Save and close the file.

A Long (Failover) Timeout

When this timeout period expires, the CM returns a PIN_ERR_TIMEOUT error to the client. In a high-availability system with multiple DMs configured, the CM connects to the secondary DM to process the requests.

To configure the failover timeout:

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

   ```
   pcm_timeout_in_msecs milliseconds
   ```

   where *milliseconds* is the number of milliseconds in the failover timeout period.

> **Note:**
>
> The value of this entry should be larger than the value of the **pcm_suspect_timeout_in_msecs** entry.

3. Save and close the file.

7

Monitoring Data Manager Activity

Learn how to check the status of Oracle Communications Billing and Revenue Management (BRM) Data Managers (DMs) at regular intervals to monitor resource usage. You can also make inferences about the operation of the DM by checking the status at intervals and comparing the results with what you expect.

Topics in this document:

Manually Checking the Status of the DM

You can check and view the status of the DM in flist format and in a report format.

> **Note:**
>
> You can also use Oracle Enterprise Manager to check the status of the DM.

Checking the DM Status in flist Format

To check the status of the DM in flist format:

1. Go to the *BRM_home*/**sys/test** directory.
2. Enter the following commands:

   ```
   testnap
   robj - database_number /status_dm 1
   ```

   where *database_number* is the database number of the DM for which you want the status.

> **Note:**
>
> You can check the status of only one DM at a time.

BRM displays the status of the DM in flist format.

Table 7-1 describes the fields in **/status_dm**.

Table 7-1 /status_dm Object Fields

/status_dm Object Field	Description
PIN_FLD_DM_BIGSIZE	Specifies the size, in bytes, of the big part of the DM shared memory.
PIN_FLD_SM_PASSTHRU_NAME	Specifies the current value of the **dm_sm_pass_thru_obj** entry in the DM **pin.conf** file.
PIN_FLD_SM_SHMSIZE	Specifies the maximum shared memory size, in bytes, for a custom DM. **Note:** Ignore this field if your system uses Oracle DM.
PIN_FLD_TRANS_OP_QUEUED	Specifies the number of transactions currently queued. This is an instantaneous counter.
PIN_FLD_DM_BACKEND	Array that defines the DM back end.
PIN_FLD_FLAGS	Specifies the internal state of the DM back end. These states are used for the internal working of the Oracle DM. • DMSHM_ALIVE 0x00001000 • DMSHM_DYING 0x00002000 • DMSHM_DEAD 0x00004000 • DMSHM_INITING 0x00008000 • DMSHM_RESTARTING 0x00010000
PIN_FLD_TATTLE_TALE	This flag is reset each time you retrieve the DM status report. This allows you to see what happened since the last DM report. • DM_TATTLE_SELECT 0x0001 • DM_TATTLE_CMD 0x0002 • DM_TATTLE_IO_IN 0x0004 • DM_TATTLE_IO_OUT 0x0008 • DM_TATTLE_ALL (DM_TATTLE_SELECT | DM_TATTLE_CMD | DM_TATTLE_IO_IN | DM_TATTLE_IO_OUT)
PIN_FLD_DM_FRONTEND	Array that defines the DM back end.
PIN_FLD_FLAGS	Specifies the internal state of the DM front end. These states are used for the internal working of the Oracle DM. • DMSHM_ALIVE 0x00001000 • DMSHM_DYING 0x00002000 • DMSHM_DEAD 0x00004000 • DMSHM_INITING 0x00008000 • DMSHM_RESTARTING 0x00010000

Table 7-1 (Cont.) /status_dm Object Fields

/status_dm Object Field	Description
PIN_FLD_TATTLE_TALE	This flag is reset each time you retrieve the DM status report. This allows you to see what happened since the last DM report. • DM_TATTLE_SELECT 0x0001 • DM_TATTLE_CMD 0x0002 • DM_TATTLE_IO_IN 0x0004 • DM_TATTLE_IO_OUT 0x0008 • DM_TATTLE_ALL (DM_TATTLE_SELECT | DM_TATTLE_CMD | DM_TATTLE_IO_IN | DM_TATTLE_IO_OUT)
PIN_FLD_CONNECTS	Specifies the number of concurrent connections the front end has received. This is an instantaneous counter.
PIN_FLD_HIWAT	Specifies the maximum number of concurrent connections the front end received during the life of the DM. This is the maximum value reached by PIN_FLD_CONNECTS for this front end.
PIN_FLD_DM_FE_CONNECT	Array that defines the front-end connection.
PIN_FLD_FLAGS	Specifies the internal state for a DM context: • DM_FLAGS_OPEN 0x00000001 • DM_FLAGS_LOST_MSG_SENT 0x00000004 • DM_FLAGS_DOING_TRANS 0x01000000 • DM_FLAGS_DIED 0x00000008 • DM_FLAGS_HEAP_GOT 0x00000010 • DM_FLAGS_SOFT_SHUTDOWN 0x00001000 • DM_FLAGS_SHUTDOWN 0x00002000
PIN_FLD_DM_FE_STATE	Specifies the current front-end state in the DM context. • **0**: Waiting to receive an operation from the CM. • **1**: Receiving an operation from the CM. • **2**: Sent an operation to be processed and waiting for the back end. • **3**: The operation is complete. • **4**: Sending a response to the CM.
PIN_FLD_DM_BE_STATE	Specifies the current back-end state in the DM context: • **1**: Busy; currently doing an operation. • **2**: Locked to a transaction.
PIN_FLD_DM_BE_IDX	Specifies the back-end index that is performing this connection (transaction).
PIN_FLD_DM_BACKEND	Array that defines the DM back end.
PIN_FLD_OPCODE	Specifies the number of the opcode that is being run. **Note:** To find an opcode's number, see the opcode header files in the *BRM_home***/include/ops** directory.
PIN_FLD_DM_USED	Specifies the memory, in bytes, dedicated to a DM context.
PIN_FLD_DM_LOW	Specifies the smallest available free memory, in bytes, in the connection's heap.
PIN_FLD_DM_HIGH	Specifies the largest available free memory, in bytes, in the connection's heap.
PIN_FLD_DM_BIG	Specifies how much big memory this connection has allocated.

Checking the DM Status in Report Format

To check the status of the DM in report format:

1. Find the process ID (PID) of the DM master process by looking in the **pid** file for the DM in *BRM_home*/**sys/dm_oracle**.
2. Enter the following command:

   ```
   kill -USR1 PID_of_DM
   ```

   where *PID_of_DM* is the process ID of the DM master process.

BRM displays the status of the DM in the **dm_oracle.log** file. The log file shows information about the DM, such as the PID, memory usage, transaction queue, and information about the back ends and the front ends.

Monitoring DM Shared Memory Usage

You can check shared memory usage by looking in the master overview section of the DM report. The number of used and free heap blocks (**# used** and **# free**) shows memory usage, expressed in 8-KB blocks. To prevent failures associated with insufficient memory, verify that **# free** is a relatively large number. If **# free** is a small portion of **# used**, you should increase the size of the shared memory area. Otherwise, operations might fail, returning PIN_ERR_NO_MEM.

Monitoring DM Transaction Queues

To check the status of transactions, look in the master overview section of the DM report. The **trans_op_cnt** entry shows the number of transactions currently being processed, and the **trans_op_queued** entry shows the number waiting to be processed. For applications that require rapid response times, you can adjust the load on the system to keep to a minimum the number of transactions waiting to be processed. See "Improving Data Manager and Queue Manager Performance".

Monitoring DM Back Ends

You can use the back-end report to identify each back-end process ID (PID), the back-end status, and the number of operations processed. A value of **0x1000** (4096) for FLAGS shows that the back end is active. The report also gives information on resource usage.

The second flag value is reset each time the DM status report is received. Therefore, you can tell what has happened (at least once) since the last DM report by a flag bit being clear.

Table 7-2 shows the flag bit values:

Table 7-2 DM Flag Bit Values

Value	Flag	Description
0x8	IO output	Never cleared for back ends.

Table 7-2 (Cont.) DM Flag Bit Values

Value	Flag	Description
0x4	IO input	Cleared when the back end starts an operation.
0x2	CMD	Cleared when the back end is given a command or transaction.
0x1	SELECT	Cleared when the back end wakes up using **select(2)**.

On a quiet back end, the second flag value stays at **f**. The counters of most interest are those that keep track of the total number of operations and total transactions.

As shown in Table 7-3, the back-end state values are a bit mask flag:

Table 7-3 Back-End State Bit Mask Values

Value	Description
0x1	Busy; currently doing an operation.
0x2	Locked to a transaction.

The back-end index and operation may be left over from the previous operation and may be no longer valid. The **used** field indicates memory usage. When idle, one 8-KB chunk is normally used. During an operation or transaction, this amount varies.

Monitoring DM Front Ends

You can use the front-end report to identify each front-end process ID (PID), the front-end status, and the number of operations processed. A value of **0x1000** (4096) for FLAGS shows that the front end is active.

For each connection, the report also gives a snapshot of the connection status. When idle, the state values should each be **0** (zero).

Table 7-4 describes the front-end state values:

Table 7-4 Front-End State Values

Value	Description
0	Waiting to receive an operation from the CM.
1	Receiving from the CM.
2	Sent an operation to be processed, waiting for back end.
3	The operation is done.
4	Sending a response to the CM.

The front-end flags are the same as the back-end flags, except that the front ends clear the IO output value when they send a reply back to the CM. The information in the connection report is a snapshot of the connection status.

Increasing the Level of Reporting for a DM

By default, DMs report errors and warnings. You can have a DM report debugging messages as well.

You can specify which debugging messages you want written to the log. The following debug settings control which debugging information is logged:

- DM_DEBUG controls which opcode-processing debug messages are logged.
- DM_DEBUG2 controls which data dictionary processing debug messages are logged.
- DM_DEBUG3 controls which SQL statement debug messages are logged based on which parts of the DM produced the SQL statements.
- DM_DEBUG5 controls which queuing debug messages are logged for the Oracle DM (**dm_oracle**) and the Synchronization Queue DM (**dm_aq**).

The *BRM_home*/**include/dm_debug.h** file contains definitions of the flags you can enable for each debug setting. You enable multiple flags for one debug setting by summing the values of the flags and including the sum in the debug setting's corresponding DM configuration (**pin.conf**) file entry (see "Editing the Configuration File to Set Debug Logging Options") or environment variable (see "Using Environment Variables to Set Debug Logging Options").

For example, to log information about transaction tracing, set DM_DEBUG to **0x70**, which is the sum of the following flags:

```
DM_DEBUG_TRANS_IN_PR     0x10
DM_DEBUG_TRANS_OUT_PR    0x20
DM_DEBUG_TRANS_TRACE     0x40
```

Depending on what information you want to log, you can include values for any combination of the debug settings (DM_DEBUG, DM_DEBUG2, DM_DEBUG3, and DM_DEBUG5).

The way you increase the level of reporting depends on your operating system and the DM:

- For all DMs other than **dm_oracle** and **dm_tax**, you can include debug settings in the DM's configuration (**pin.conf**) file. Specify each debug setting separately as in the following example:

  ```
  - dm dm_debug    0xFFF003FF
  - dm dm_debug2   0x10
  - dm dm_debug3   0x10
  ```

  See "Editing the Configuration File to Set Debug Logging Options".

- For **dm_oracle** and **dm_tax**, you must specify the debug settings as environment variables. Set a separate environment variable for each debug setting.

  See "Using Environment Variables to Set Debug Logging Options".

Editing the Configuration File to Set Debug Logging Options

To set debug logging options for all DMs except **dm_oracle** and **dm_tax**:

1. Stop the DM. See "Starting and Stopping the BRM System".
2. Open the configuration file (**pin.conf**) for this DM.
3. Edit the debug setting entries to set the level of debugging reporting.
4. Save and close the file.
5. Start the DM. See "Starting and Stopping the BRM System".
6. Run the DM operations for which you want debugging information.
7. Stop the DM.
8. Open the log file for the DM (for example, **dm_fusa.pinlog**) and review the messages.
9. Return DM logging to its normal level by commenting out the debug setting entries in the configuration file. Otherwise, subsequent DM activity will generate large log files.

Using Environment Variables to Set Debug Logging Options

To set debug logging options for **dm_oracle** and **dm_tax**:

1. Stop the DM. See "Starting and Stopping the BRM System".
2. In the environment from which the DM starts, set an environment variable for each debug setting. For example:

   C shell:

   ```
   setenv DM_DEBUG3 0xFFFF003F
   ```

   Korn shell:

   ```
   DM_DEBUG3=0xFFFF003F
   export DM_DEBUG3
   ```
3. Start the DM. See "Starting and Stopping the BRM System".
4. Run the DM operations for which you want to display debugging information.
5. Stop the DM.
6. Open the log file for the DM (for example, **dm_oracle.pinlog**) and review the messages.
7. Return DM logging to its normal level. Otherwise, subsequent DM activity will generate large log files.

Logging the DM Process Time Information for Performance Diagnostics

To diagnose performance problems with the DM process, you can configure the DM to log the time it takes to process each opcode. You can use this information to determine the time the DM spends on its internal operations and the time it spends on the database operations.

Before the DM starts processing an opcode, it logs the current time. Then for each SQL statement that the DM sends to the database for the opcode, it logs the following information:

- Session ID
- Statement ID
- Time taken by the database to process the SQL statement

To log the timing information for the SQL statement, set the DM_DEBUG3 flag to **0x00010000**, which corresponds to the DM_DEBUG3_TIME_INFO variable defined in the *BRM_home*/**include/dm_debug.h** file.

Replacing Failed DM Child Processes

All DMs, such as Paymentech DM and Email DM are set to automatically replace child processes that have stopped. This feature prevents the system from losing DM processes as a result of transient failures over time. For initial testing, or if you have recurring errors that would cause a “fork and die" endless loop (in an Oracle database, for example), you can tell the DM to not replace failed child processes:

1. Open the configuration file (**pin.conf**) for this DM.
2. Change the value of the **dm_restart_children** entry to **0**.
3. Save and close the file.
4. Stop and restart the DM.

When a child process stops and is replaced, BRM notes the event in the error log file for the DM.

> **Note:**
>
> BRM does not automatically replace child processes that are hung. See "Problem: Hung and or Looping Processes".

8
Getting Quality of Service Statistics

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) to collect opcode performance quality of service (QoS) statistics.

Topics in this document:

See also "Monitoring Your BRM System" and "Monitoring Connection Manager Activity".

Getting Quality of Service Statistics

You can collect statistics about CM opcode performance (for example, the number of times an opcode is called or the number of times an opcode returns an error). You can collect statistics on a per-opcode basis. The statistics are written to the CM log file whenever a client connection closes. You can enable and disable this feature by modifying the CM **pin.conf** file.

To measure latency for an opcode, you can specify up to seven maximum latency times, with each latency time period representing a *QoS bucket*. For example, if you specify latencies of 10, 20, and 100, the buckets are:

- 0-10 milliseconds: QoS bucket 1
- 10-20 milliseconds: QoS bucket 2
- 20-100 milliseconds: QoS bucket 3
- Greater than 100 milliseconds: QoS bucket 4

The QoS buckets are defined as follows:

- QoS bucket 1: less than or equal to QoS time 1
- QoS bucket 2: greater than QoS time 1 and less than or equal to QoS time 2
- QoS bucket 3: greater than QoS time 2 and less than or equal to QoS time 3
- QoS bucket 4: greater than QoS time 3

The information listed in Table 8-1 is collected per opcode:

Table 8-1 Quality Service Statistics from the CM

Statistic	Description
Opcode	The opcode that this information pertains to.
Interval timestamp	The starting timestamp of this interval.
Total opcode call count	The number of times this opcode has been called in this time interval.
Total error count	The number of times this opcode has returned an error in this time interval.
Minimum latency	The fastest elapsed time that this opcode took to complete without returning an error.

Table 8-1 (Cont.) Quality Service Statistics from the CM

Statistic	Description
Timestamp of minimum latency	The timestamp when the minimum latency occurred.
Maximum latency	The slowest elapsed time that this opcode took to complete without returning an error.
Timestamp of maximum latency	The timestamp when the maximum latency occurred.
Total latency	Total latency of all successful calls to this opcode, not including the calls that returned an error.
Input flist of maximum latency	The input flist that was used when the maximum latency occurred.
QoS bucket count	The number of active QoS buckets for this opcode.
QoS bucket 1 counts	The number of times that the latency of a successful call to the opcode falls into each bucket. For example, 10 in bucket 1, 12 in bucket 2, and so forth.
QoS bucket times2 count	The maximum time in nanoseconds for each QoS bucket.
Timestamp of first received opcode	The timestamp when the first opcode was received.
Timestamp of last received opcode	The timestamp when the latest opcode was received.

Configuring CM QoS Statistics

To enable or disable the collection of opcode QoS statistics:

1. Open the CM **pin.conf** file in *BRM_home*/**sys/cm**.
2. Change the value of the **cm_opcode_stats** entry.

   The syntax is as follows:

   ```
   - cm cm_opcode_stats opcode QoS_1 [, QoS_2, ... QoS_7]
   ```

   where *opcode* can be an opcode name or opcode number.

   For example, to use an opcode name and four buckets, enter the following:

   ```
   - cm cm_opcode_stats PCM_OP_CUST_COMMIT_CUSTOMER 10, 20, 30
   ```

   For example, to use an opcode number and four buckets, enter the following:

   ```
   - cm cm_opcode_stats 63 10, 20, 30
   ```

   > **Note:**
   >
   > If the entry does not exist, you can add it anywhere in the file.

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

9

Collecting Diagnostic Information by Using RDA

Learn how to use Remote Diagnostic Agent (RDA) to monitor your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Collecting Diagnostic Information by Using RDA

Remote Diagnostic Agent (RDA) is an Oracle standard tool used to collect diagnostic data from your system applications environment.

> **Note:**
>
> RDA replaces the Support Informer utility. Support Informer is obsolete and no longer supported. However, Support Informer libraries continue to be packaged with BRM. The libraries are accessed by the RDA profile named **SupportInformer** at run time.

Use RDA to collect information about your BRM system. When you submit a service request (SR) to Oracle Technical Support, you must also provide an RDA output file. The RDA output file provides a comprehensive view of your system configuration and contains diagnostic data used by Oracle Technical Support to diagnose problems. This minimizes the number of requests from Oracle Technical Support for additional information, which can reduce the service request resolution time.

You can use RDA to collect BRM diagnostic information. The information collected from BRM includes:

- Component log files

  RDA collects component log data from the component **.pinlog**, **.log**, and **Infranet.properties** files. For example, RDA collects the log data for BRM invoice formatter from **formatter.pinlog**, **formatter.log**, and **Infranet.properties**.

- Application log files

  RDA collects application log data from the application **.pinlog**, **.log**, and **Infranet.properties** files. For example, RDA collects the log data for Batch Controller from **batch_controller.pinlog**, **BatchController.log**, and **Infranet.properties**.

- Configuration files

  RDA collects configuration data from the **pin.conf** file. For example, RDA collects CMMP configuration data from the CMMP **pin.conf** file.

- Other files

  RDA collects installation and version details from the **vpd.properties** and *Oracle_home***/inventory/registry.xml** files.

To find BRM component information, RDA looks in the *BRM_home***/RDA** directory.

RDA collects the following customer-specific information:

- Company name
- Contact person
- Contact email
- Comment on the collection
- Service request (when applicable)

> **Caution:**
>
> When you run **rda.sh**, the script returns the "Perl not found in the PATH" error and the command fails. To work around this issue, remove the **.config** file (hidden file) in the RDA directory. Oracle recommends that you do not use shell script for RDA.

Installing Remote Diagnostic Agent

You can use the RDA JAR file packaged with the BRM software. The RDA JAR file is available in the *BRM_home***/RDA** directory. You can extract and use this JAR file to run RDA.

To extract the RDA JAR file:

1. Go to the *BRM_home***/RDA** directory.
2. Run the following command:

```
jar -xvf RDA.jar
```

> **Note:**
>
> - RDA is not supported on Windows.
> - RDA collects diagnostic and configuration data for all BRM components and applications *only* from the server on which RDA is running. To collect data for BRM components and databases on other servers, install and run RDA on the other servers.

To determine whether RDA is installed on a server, run the following command:

```
perl rda.pl -cv
```

If RDA is installed on the server without any error, the following message is displayed: “No issues found in the RDA installation."

RDA includes the **SupportInformer** profile, which runs the following modules:

- **S380BRM**

  Collects Oracle Communications BRM information.
- **S105PROF**

  Collects the user profile data.
- **S110PERF**

  Collects performance information.
- **S100OS**

  Collects operating system information.

> **Note:**
>
> In addition to the preceding modules, the RDA profile runs other modules, such as INI, CFG, END, RDSP, and LOAD.

Running Remote Diagnostic Agent

To run RDA:

1. Go to the *BRM_home*/**RDA** directory and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```
2. To collect system information, verify that the PIN_HOME environment variable is set to the BRM installation directory.
3. To run RDA, you must first perform an initial setup and then run data collection. To perform the initial setup, run the following command:

   ```
   perl rda.pl -S
   ```
4. Run one or more of the following commands:
   - To identify the list of modules:

     ```
     perl rda.pl -L m
     ```
   - To identify the list of profiles:

     ```
     perl rda.pl -L p
     ```
   - To identify the list of modules for the available profiles:

     ```
     perl rda.pl -x profiles
     ```
   - To perform BRM data collection using default values:

     ```
     perl rda.pl -v
     ```

> **Note:**
>
> To collect database-specific data, you must run the command as a SYSDBA because DBA privileges are required to collect the database tables data.

When you run RDA, it prompts for information to determine what data to collect and for which products. You can choose to accept the default values or change them based on your BRM system configuration. RDA saves all your responses to the **/rda/setup.cfg** file.

For example, to initialize data collection and to generate the output files, RDA prompts for the following setup information:

```
S000INI: Initializes the Data Collection

Enter the prefix for all the files generated
Hit 'Return' to accept the default (RDA)
>
Enter the directory used for all the files generated
Hit 'Return' to accept the default (/rda/output)
>
Do you want to keep report packages from previous runs (Y/N)?
Hit 'Return' to accept the default (N)
>
Is a fresh collection done as default action (Y/N)?
Hit 'Return' to accept the default (Y)
>
Enter the Oracle Home to be used for data analysis
Hit 'Return' to accept the default
>
Enter the domain name of this server
Hit 'Return' to accept the default (portal.com)
>
```

You can also run RDA in noninteractive mode by using command-line options:

rda.pl -v -d -S -C -R -P -p *profile_name* [*-db_version*]

- **-v**: Set verbose mode
- **-d**: Set debug mode
- **-S**: Set up specified modules
- **-C**: Collect diagnostic information
- **-R**: Generate specified reports
- **-P**: Package the reports
- **-p** *profile_name* [*-db_version*]: Specify the setup profile and the database version. The database version is used only to collect database-specific data.

To collect BRM diagnostic data, run the following command:

```
perl rda.pl -vdSCRP -p SupportInformer
```

To collect BRM- and database-specific data, run the following command:

```
perl rda.pl -vdSCRP -p SupportInformer-DB12c
```

> **Note:**
>
> The database version in the **SupportInformer** profile depends on the version of the database installed for BRM.
>
> See "BRM Software Compatibility" in *BRM Compatibility Matrix* for the version of the database.

The final output is packaged in an archive located in the output directory chosen during RDA setup. RDA output is not encrypted and can be viewed by anyone using any Web browser.

For information on reporting RDA problems, see "Contacting Technical Support".

10

Monitoring Key Performance Indicators

Learn how to monitor key performance indicators (KPIs) in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

See also "Monitoring Your BRM System" and "Getting Quality of Service Statistics".

Using the pin_db_alert Utility to Monitor Key Performance Indicators

Key performance indicators (KPIs) are metrics you use to quantify the health of your database and to alert you when potential issues exist. They identify database tables that must be archived or purged and indexes, triggers, and stored procedures that are missing or invalid.

KPIs are monitored when you run the **pin_db_alert.pl** utility. Generally you set up a **cron** job to run the utility periodically to monitor the health of your database. For more information, see "Running the pin_db_alert.pl Utility".

Each KPI is identified by an ID that associates a component being monitored to a corresponding validation value. For example, you can monitor the size of an audit table with a size threshold that monitors the number of rows in that audit table. When the threshold value is reached, the results are returned and an alert notification can be sent, warning you of the component's condition.

The component and validation functionality for each KPI comprises:

- A data extraction module, which queries the database for the KPI data and writes the results to an output file.
- A validation module, which compares the query results to validation parameters defined in a configuration file and writes the validation status to an output file.

After the validation results are written to the output files, a decision module (**DecisionUtility.pm**) evaluates each KPI result and determines whether to generate email alert notifications based on the KPI result status. For more information, see "About KPI Status and Alert Notifications".

KPI Default Behavior

Table 10-1 contains a list of supported KPIs and provides the default behavior of their data and validation modules.

Table 10-1 Supported KPIs and Default Behavior

KPI ID	Default Behavior
AuditHistoryAge	The **auditAge** module calculates the age of the audit tables listed in the **pin_db_alert.conf** file's DATA_PLUGINS entry and DEFAULT_AUDIT_TABLES entry. It writes the results to the **auditAge_AuditHistoryAge.out** file. The **auditAge_validation** module uses threshold values in the **auditAge** validation configuration file to determine which audit tables in the results file are at the threshold, and writes them to the **auditAge_validation_AuditHistoryAge.out** file. For information on changing the default age thresholds, see "Monitoring the Age of Audit Tables".
AuditTableSize	The **auditSize** module calculates the number of rows present in the audit tables listed in the **pin_db_alert.conf** file's DATA_PLUGINS entry and DEFAULT_AUDIT_TABLES entry and writes the results to the **auditSize_AuditTableSize.out** file. The **auditSize_validation** module uses the threshold values in the **auditSize** validation configuration file to determine which audit tables in the results file are at the threshold and writes them to the **auditSize_validation_AuditTableSize.out** file. For information on changing the default size thresholds, see "Monitoring the Size of Audit Tables".
OldestEventAge	The **eventData** module calculates the age of the oldest event in the **event_t** table, as well as the records in the tables defined in the **pin_db_alert.conf** file's DATA_PLUGINS entry, and writes the results to the **eventData_OldestEventAge.out** file. The **eventData_validation** module uses the threshold values in the **eventData** validation configuration file to determine which entries in the results file are at the threshold, and writes them to the **eventData_validation_OldestEventAge.out** file. For information on changing the default event age, see "Monitoring the Age of Events".
ACTIVETRIGGERS	The **triggersList** module retrieves a list of active triggers in the BRM system and writes their names and status (ENABLED or DISABLED) to the **triggersList_ACTIVETRIGGERS.out** file. The **triggersList_validation** module compares the list of active triggers in the **triggersList** validation configuration file to the triggers in the results file and writes missing triggers to the **triggersList_validation_ACTIVETRIGGERS.out** file. **Important:** If you installed optional managers that use unique triggers or if you created custom triggers, you must add them to the **triggersList** validation configuration file to monitor their status. See "Monitoring Active Triggers".
INDEXES	The **indexList** module retrieves a list of unique indexes in the BRM system and writes the index names and uniqueness values to the **indexList_INDEXES.out** file. The table name and column name for each index is also listed. The **indexList_validation** module compares the list of indexes in the **indexList** validation configuration file to the indexes in the results file and writes missing or invalid indexes to the **indexList_validation_INDEXES.out** file. **Important:** If you installed optional managers that use unique indexes or if you created custom indexes, you must add them to the **indexList** validation configuration file to monitor their status. See "Monitoring Indexes".

Table 10-1 (Cont.) Supported KPIs and Default Behavior

KPI ID	Default Behavior
PROCEDURES	The **proceduresList** module retrieves a list of stored procedures in the BRM system and writes the stored procedure names and status (VALID or INVALID) to the **proceduresList_PROCEDURES.out** file. The **proceduresList_validation** module compares the list of stored procedures in the **proceduresList** validation configuration file to the procedures in the results file and writes missing procedures to the **proceduresList_validation_PROCEDURES.out** file. **Important:** If you installed optional managers that use unique stored procedures or if you created custom stored procedures, you must add them to the **proceduresList** validation configuration file to monitor their status. See "Monitoring Stored Procedures".

You can enable email alerts to notify a list of people about the validation results. For more information, see "Setting Up Email Alert Notifications".

About KPI Status and Alert Notifications

When the **pin_db_alert.pl** utility runs, it returns a PASS or FAILURE status for each configured KPI, which includes a severity level for the status. The following severity levels listed in Table 10-2 are possible for any KPI:

Table 10-2 KPI Status Severity Levels

Severity	Description
CRITICAL	Performance, functionality, or both are heavily impacted and require immediate attention. Critical failures generally involve data corruption (for example, when an event table is missing data after a system upgrade). Set up alert notifications for critical failures so you can correct such problems immediately and avoid further corruption.
MAJOR	Performance, functionality, or both are impacted and require immediate attention. Major failures generally involve potentially serious performance degradations (for example, when an index is missing or an index contains columns that are out of order). These problems can occur when you customize your BRM software. Major failures also include issues where functionality can be impacted. For example, if the TRIG_CYCLE_DEFERRED_TAX trigger is missing and billing runs, cycle taxes will not be calculated. Set up alert notifications for major failures so you can correct problems immediately and avoid further degradation or data corruption.
MINOR	Performance might be impacted and will need attention in the future. Minor failures involve large audit tables.
WARNING	Performance and functionality both work as expected, but performance may be impacted in the future. For example, depending on your hardware and software resources, you can set up an alert notification when an event table reaches an age threshold or an audit table reaches a size threshold, so they can be archived or purged. Warning failures generally do not impact performance and *never* impact functionality.
NORMAL	No data or performance risks were found. This status is valid only for PASS results.

You can configure the **pin_db_alert.pl** utility to send email notifications to alert a list of people when a KPI is at a specified severity level. For more information, see "Setting Up Email Alert Notifications".

Setting Up KPI Monitoring

To monitor KPIs, first you configure the KPI data entries in the **pin_db_alert.pl** utility's configuration file, and then you set up the validation thresholds in each validation module's configuration file.

The **pin_db_alert.pl** utility's configuration file contains entries for all KPIs; therefore, Oracle recommends that you configure this file for all KPIs *before* you set up the validation thresholds for each individual KPI.

> **Note:**
>
> If you do not define KPI validation thresholds, the validation process will not occur; therefore, any alert notifications you configured will not be sent.

Setting up KPI monitoring requires the following:

- Specifying Which KPI Data Is Extracted
- Setting Up Email Alert Notifications
- Enabling Database Access
- Monitoring the Size of Audit Tables
- Monitoring the Age of Audit Tables
- Monitoring the Age of Events
- Monitoring Active Triggers
- Monitoring Indexes
- Monitoring Stored Procedures

The default configuration for monitoring KPIs is defined in the **pin_db_alert.pl** utility's configuration file (*BRM_home***/diagnostics/pin_db_alert/pin_db_alert.conf**).

To edit this file, open it with a text editor and perform the following tasks as necessary. For more information, see the comments in the **pin_db_alert.conf** file.

- In the KPI_IDS entry, specify the KPI ID for each KPI to monitor.

  By default, all KPIs are listed; therefore, if you do not want to monitor one, remove it from the default list. For a list of KPI IDs, see "KPI Default Behavior".

- In the DATA_PLUGINS entry, specify the data module and desired values for each KPI listed in the KPI_IDS entry. See "Specifying Which KPI Data Is Extracted".

> **Note:**
>
> In the sample **pin_db_alert.conf** file, values are provided for the **AuditHistoryAge** and **AuditTableSize** KPIs; however, the **OldestEventAge** KPI does not contain any values. You must provide your own values. See "Monitoring the Age of Events".

- In the VALIDATION_PLUGINS entry, specify the validation module for each KPI listed in the KPI_IDS entry.

> **Note:**
>
> Make sure the validation modules are listed in the same order as their associated data modules in the DATA_PLUGINS entry.

- In the STATUS entry, configure the alert notifications. Specify the status and severity, and list the email addresses that get notified by the status/severity combination. For more information, see "Setting Up Email Alert Notifications".
- In the DEFAULT_AUDIT_TABLES entry, specify which audit tables to monitor by default. These audit tables are monitored in addition to any tables you list as values in the DATA_PLUGINS entry for the **auditAge** and **auditSize** modules.
- In the DB_USER and DB_PASSWD entries, specify the database user ID and encrypted password that are listed in the **sm_id** and **sm_pw** entries in the Data Manager (DM) **pin.conf** file. For more information, see "Enabling Database Access".

Specifying Which KPI Data Is Extracted

To specify which data is extracted from the database during KPI monitoring:

1. Open the **pin_db_alert.pl** utility's configuration file (*BRM_home*/**diagnostics/pin_db_alert/pin_db_alert.conf**) with a text editor.
2. In the DATA_PLUGINS entry, specify the data module and desired values for each KPI in the KPI_IDS entry:
   - **To extract data for the auditAge data module:**

     Specify the audit table names to monitor using the following syntax, separating each audit table name by a space:

     ```
     @DATA_PLUGINS =("auditAge Audit_table_name  Audit_table_name");
     ```

     > **Note:**
     >
     > These tables are in addition to audit tables you have listed in the DEFAULT_AUDIT_TABLES entry.

   - **To extract data for the auditSize data module:**

     Specify the audit table names to monitor using the following syntax, separating each audit table name by a space:

```
@DATA_PLUGINS =("auditSize Audit_table_name  Audit_table_name");
```

> **Note:**
>
> These tables are in addition to audit tables you have listed in the DEFAULT_AUDIT_TABLES entry.

- **To extract data for the eventData module:**

  Specify the events to monitor using the following syntax:

  ```
  @DATA_PLUGINS =("eventData
  Table_name:Column_name:Operator:Column_value");
  ```

  where:

  - *Table_name* is the name of the table that contains the event data.
  - *Column_name* is the name of the table column that contains the event data.
  - *Operator* is any standard SQL operator.
  - *Column_value* is the POID of the event.

  For example:

  ```
  @DATA_PLUGINS =("eventData event_t:account_obj_id0:=:21950");
  ```

  > **Note:**
  >
  > You can add any number of values for the **eventData** module, separated by spaces; however, you can specify only one operator per table. If the operator or syntax is incorrect, the table is not validated, and an error is written to the data extraction output file.

- **To extract data for the triggersList, proceduresList, and indexList modules:**

  The **triggersList**, **proceduresList**, and **indexList** modules take no values. To extract data for these modules, list them in the DATA_PLUGINS entry using the following syntax:

  ```
  @DATA_PLUGINS =("triggersList","proceduresList","indexList");
  ```

  Enclose the entire DATA_PLUGINS value string with parentheses ( ) and separate each data value string with commas. For example:

  ```
  @DATA_PLUGINS =("auditSize au_service_t au_product_t au_account_t
  au_rate_t",
  "eventData event_t:account_obj_id0:=:21956 account_t:poid_id0:=:21956:",
  "auditAge au_service_t au_product_t",
  "triggersList","proceduresList","indexList");
  ```

3. Save and close the file.

Setting Up Email Alert Notifications

To configure the **pin_db_alert.pl** utility to send email notifications when a KPI validation returns a specified result/severity combination:

1. Open the **pin_db_alert.pl** utility's configuration file (*BRM_home*/**diagnostics/pin_db_alert.conf**) with a text editor.
2. Edit the **STATUS** entry using the following syntax:

   ```
   'Error:MAIL_ALERT:Notification_list'
   ```

   where:

   - *Error* is a combination of the status and severity, separated by a dot (**.**). The following values are valid:

     FAIL.CRITICAL

     FAIL.MAJOR

     FAIL.MINOR

     FAIL.WARNING

     PASS.WARNING

     PASS.NORMAL

   - *Notification_list* is a comma-separated list of email addresses to which the validation results are sent. You can have any number of email addresses for any error.

   Be sure to enclose each status string in single quotation marks (' ').

   For example:

   ```
   @STATUS=('FAIL.CRITICAL:MAIL_ALERT:IT@example.com', 'FAIL.MINOR:MAIL_ALERT:
   example@example.com, sysadm@example.com');
   ```

   > **Note:**
   >
   > You cannot configure email alerts for a specific KPI.

3. Save and close the file.

Enabling Database Access

The **pin_db_alert.pl** utility requires the database user name and password to query the database for KPIs.

1. Open the **pin_db_alert.pl** utility's configuration file (*BRM_home*/**diagnostics/pin_db_alert/pin_db_alert.conf**).
2. In the DB_USER and DB_PASSWD entries, specify the database user ID and encrypted password, respectively.

> **Note:**
>
> These must be the same database user ID and password specified in the **sm_id** and **sm_pw** entries in the DM **pin.conf** file.

Use the following syntax:

```
DB_USER="User_ID";
DB_PASSWD="Encrypted_passwd";
```

3. Save and close the file.

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

Monitoring the Size of Audit Tables

To monitor the size of audit tables:

1. If necessary, specify the **auditSize** module values in the DATA_PLUGINS entry of the **pin_db_alert.pl** utility's configuration file. See "Setting Up KPI Monitoring".
2. Open the **auditSize** validation configuration file (*BRM_home*/**diagnostics/pin_db_alert/auditSize_validation_AuditTableSize.conf**) with a text editor.
   - To change a size threshold for an existing table, change the number of rows specified in the AUDIT_SIZE_THRESHOLD value for that table.
   - To add an audit table, add a new AUDIT_SIZE_THRESHOLD entry for that table.
   - To omit an audit table from the validation process, either delete the AUDIT_SIZE_THRESHOLD entry for that table or comment out the entry.

   For details on how to configure the AUDIT_SIZE_THRESHOLD entry, see the comments in the **AuditTableSize** configuration file.
3. Save the file.

Monitoring the Age of Audit Tables

To monitor the age of audit tables:

1. If necessary, specify the **auditAge** module values in the DATA_PLUGINS entry of the **pin_db_alert.pl** utility's configuration file. See "Setting Up KPI Monitoring".
2. Open the **auditAge** validation configuration file (*BRM_home*/**diagnostics/pin_db_alert/auditAge_validation_AuditHistoryAge.conf**) with a text editor.
   - To change an age threshold for a table, change the number of days specified in the AUDIT_AGE_THRESHOLD value for that table.
   - To add an audit table, add a new AUDIT_AGE_THRESHOLD entry.
   - To omit an audit table from the validation process, either delete the AUDIT_AGE_THRESHOLD entry for that table or comment out the entry.

   For details on how to configure the AUDIT_AGE_THRESHOLD entry, see the comments in the **AudiHistoryAge** configuration file.

3. Save the file.

Monitoring the Age of Events

To monitor the age of events:

1. If necessary, configure the **eventData** module values in the DATA_PLUGINS entry of the **pin_db_alert.pl** configuration file (*BRM_home*/**diagnostics/pin_db_alert.conf**). See "Specifying Which KPI Data Is Extracted".

   > **Note:**
   >
   > You can add any number of arguments for the **eventData** module; however, you can specify only one operator per table. If the operator or syntax is incorrect, the table is not validated, and an error is written to the data extraction output file.

2. Open the **eventData** validation configuration file (*BRM_home*/**diagnostics/pin_db_alert/eventData_validation_OldestEventAge.conf**) with a text editor.
   - To change an age threshold, change the number of days specified in the OLDEST_THRESHOLD value for the table.
   - To add a table to monitor, add a new OLDEST_THRESHOLD entry for the table.
   - To omit a table from the validation process, either delete the OLDEST_THRESHOLD entry for that table or comment it out.

   For details on how to configure the OLDEST_THRESHOLD entry, see the comments in the **OldestEventAge** configuration file.
3. Save the file.

Monitoring Active Triggers

To monitor a trigger for an optional manager or customization that is not part of BRM:

1. If necessary, specify the **triggersList** module in the DATA_PLUGINS entry in the **pin_db_alert.pl** utility's configuration file. See "Setting Up KPI Monitoring".
2. Open the ACTIVETRIGGERS validation configuration file (*BRM_home*/**diagnostics/pin_db_alert/triggersList_validation_ACTIVETRIGGERS.conf**) with a text editor.
3. Add a new entry for the trigger using the following syntax:

   ```
   ENABLED trigger_name
   ```
4. Save the file.
5. Restart the Connection Manager (CM).

Monitoring Indexes

To monitor an index for an optional manager or customization that is not part of BRM:

1. If necessary, specify the **indexList** module in the DATA_PLUGINS entry in the **pin_db_alert.pl** utility's configuration file. See "Setting Up KPI Monitoring".

2. Open the *BRM_home*/**diagnostics/pin_db_alert/indexList_validation_INDEXES.conf** file.
3. Add a new entry for the index using the following syntax:

   ```
   table_name   column_name   index_name   UNIQUE
   ```

   To add a composite index, add each column name as a separate entry, in the order of the columns in the index. For example:

   ```
   ACCOUNT_NAMEINFO_T    OBJ_ID0      I_ACCOUNT_NAMEINFO__I    UNIQUE
   ACCOUNT_NAMEINFO_T    REC_ID       I_ACCOUNT_NAMEINFO__I    UNIQUE
   ACCOUNT_T             ACCOUNT_NO   I_ACCOUNT_NO__ID         UNIQUE
   ```

4. Save the file.

Monitoring Stored Procedures

To monitor a stored procedure for an optional manager or customization that is not part of BRM:

1. If necessary, specify the **proceduresList** module in the DATA_PLUGINS entry in the **pin_db_alert.pl** utility's configuration file. See "Setting Up KPI Monitoring".
2. Open the PROCEDURES validation configuration file (*BRM_home*/**diagnostics/pin_db_alert/proceduresList_validation_PROCEDURES.conf** file) with a text editor.
3. Add a new entry for the stored procedure using the following syntax:

   ```
   procedure_name VALID
   ```

4. Save the file.

Running the pin_db_alert.pl Utility

Run the **pin_db_alert.pl** utility periodically to monitor the health of your database. The **cron** command is the typical way to do this.

> **Note:**
>
> You can also run the **pin_db_alert.pl** utility manually at the command line (for example, after system upgrades).

Use a **cron** job with a **crontab** entry to run the **pin_db_alert.pl** utility at a specified time. The following **crontab** entry runs the utility at 1:00 a.m. on a quarterly basis:

```
0 1 * */3 * BRM_home/bin/pin_db_alert.pl &
```

Defining Custom KPIs

You can define custom KPIs (for example, to monitor the integrity of customer subscriber information after system upgrades):

- Define a new KPI called **SubscriberInformation** to monitor the consistency of subscriber data over a period of time. This KPI must include a data module that retrieves the subscriber information and a validation module that verifies this data.
- Create a configuration file for the KPI validation module and specify the relevant threshold information.
- Add the new KPI information to the **pin_db_alert.conf** file. For information on the entries in this file, see "Setting Up KPI Monitoring".

11

Running Stored Procedures

Learn how, in special cases, you can run custom stored procedures against the Oracle Communications Billing and Revenue Management (BRM) database.

Topics in this document:

About Running Stored Procedures in BRM

You can access data in the BRM database by using the BRM base opcodes, such as PCM_OP_SEARCH, PCM_OP_READ_FLDS, PCM_OP_WRITE_FLDS, and so on. While these opcodes will meet most of your business needs, you may occasionally need to perform complex joins, aggregations, and updates on the BRM data. To do so, you can run a custom stored procedure against the BRM database.

Adding Custom Stored Procedures to BRM

If you want to run a custom stored procedure against the BRM database, you must add the stored procedure's name to the **/config/stored_procedure** object. BRM allows only the stored procedures listed in **/config/stored_procedure** to be run in BRM.

To add custom stored procedures to BRM, you edit the **config_stored_procedure.xml** file and then load it into the database by using the **load_config** utility.

To allow BRM to run custom stored procedures:

1. Open the *BRM_home***/sys/data/config/config_stored_procedure.xml** file.
2. Add a **<RESULTS>** array element for each custom stored procedure:

   ```
   <RESULTS elem="x">
      <NAME>storedProcedure</NAME>
      <DESCR>description</DESCR>
   </RESULTS>
   ```

   where *storedProcedure* is the name of your custom stored procedure, and *description* is a short description of what the stored procedure does.
3. Save and close the XML file.
4. Load the file into the database by running the **load_config** utility:

   ```
   load_config config_stored_procedure.xml
   ```

If the XML file is in a different directory from which you run the utility, include the entire path to the file.

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

To verify that your custom stored procedures were loaded into the database, display the **/config/stored_procedure** object by using the Object Browser or by using the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

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

Running Stored Procedures

To run a stored procedure against the BRM database, you use the PCM_OP_EXEC_SPROC opcode. Configure your custom client application to call the opcode, or run the opcode by using **testnap**. See "Using the testnap Utility to Test BRM" in *BRM Developer's Guide* for information about using **testnap**.

> **Caution:**
>
> - Running stored procedures using PCM_OP_EXEC_SPROC should be done with caution as there are chances for data corruption or undesired results if not used with the utmost care.
> - Stored procedures have fixed input and output parameters, which may inhibit the extensibility of a feature if not used carefully.

The input flist that you pass into the PCM_OP_EXEC_SPROC opcode must specify the stored procedure to run and include an array of PIN_FLD_ARGS elements for defining the parameters required by the stored procedure. You can also include an optional PIN_FLD_RESULTS element for defining the results to return in the output flist. For example:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /procedure -1 0
0 PIN_FLD_PROC_NAME          STR [0] "Procedure_Name"
0 PIN_FLD_ARGS             ARRAY [1]
1     PIN_FLD_ACCOUNT_OBJ   POID [0] 0.0.0.1 /account 1536772 0
1     PIN_FLD_ARG_TYPE      ENUM [0] 0
0 PIN_FLD_RESULTS          ARRAY [1]
```

These fields are used for simple input and output parameters:

- PIN_FLD_POID: Set this to a type-only POID for the **/procedure** object.
- PIN_FLD_PROC_NAME: Set this to the name of the stored procedure to run. The name of the stored procedure must be listed in the *BRM_home***/sys/data/config/config_stored_procedure.xml** file.
- PIN_FLD_ARGS: An array of input and output parameters. These must be defined in the order in which the parameters are defined in the stored procedure. There is no support for named parameters.

  - PIN_FLD_ARG_TYPE: Specify whether the field is an input (**0**) or an output (**1**) parameter.
  - PIN_FLD_ACCOUNT_OBJ: This represents any BRM field type. For POID fields, the individual component values are split out and passed as separate parameters. Thus, a single POID would map to four distinct parameters in the stored procedure.
- PIN_FLD_RESULTS: An array of parameters for the opcode to return in its output flist.

Sample of Running a Custom Stored Procedure

The following shows an example of how to run a custom stored procedure. This example includes the following high-level steps:

1. Create a custom stored procedure named **get_gl_total**.
2. Add **get_gl_total** to the **/config/stored_procedure** object.
3. Create an input flist for PCM_OP_EXEC_SPROC.
4. Use **testnap** to run **get_gl_total** through the PCM_OP_EXEC_SPROC opcode.

Creating Sample Stored Procedure

The following creates a simple stored procedure named **get_gl_total** that returns the sum of **/journal** data for an account, grouped by **resource_id** and **gl_id**.

```
CREATE OR REPLACE PROCEDURE get_gl_total (
   i_account_obj_db   IN NUMBER,
   i_account_obj_type IN VARCHAR2,
   i_account_obj_id0  IN NUMBER,
   i_account_obj_rev  IN NUMBER,
   out_cv             IN OUT SYS_REFCURSOR) AS
BEGIN
   OPEN out_cv FOR
       SELECT account_obj_db,
              account_obj_type,
              account_obj_id0,
              0 account_obj_rev,
              resource_id,
              gl_id,
              SUM(db_ar_net_amt) AS amount
         FROM journal_t
        WHERE gl_id > 0
          AND DECODE(i_account_obj_id0, 0, account_obj_id0,
i_account_obj_id0) = account_obj_id0
     GROUP BY account_obj_db,
              account_obj_type,
              account_obj_id0,
              resource_id,
              gl_id;
END;
```

Adding Sample Stored Procedure to BRM

The following procedure shows how to add **get_gl_total** to the list of stored procedures that can be run against the BRM database:

1. Add the following **<RESULTS>** array element to the *BRM_home***/sys/data/config/config_stored_procedure.xml** file:

```
<RESULTS elem="80">
   <NAME>get_gl_total</NAME>
   <DESCR>This stored procedure sums journal data for an account</
DESCR>
</RESULTS>
```

2. Load the file into the database by running the **load_config** utility:

```
load_config config_stored_procedure.xml
```

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

Creating Sample Input Flist

The following shows a sample input flist for PCM_OP_EXEC_SPROC for running the **get_gl_total** stored procedure. It specifies to retrieve journal data for account 1536772 and return the **/account** object POID, resource ID, G/L ID, and amount for each journal entry.

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /procedure -1 0
0 PIN_FLD_PROC_NAME           STR [0] "get_gl_total"
0 PIN_FLD_ARGS              ARRAY [1]
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1536772 0
1     PIN_FLD_ARG_TYPE       ENUM [0] 0
0 PIN_FLD_RESULTS           ARRAY [0]
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account -1 0
1     PIN_FLD_RESOURCE_ID     INT [0] 0
1     PIN_FLD_GL_ID           INT [0] 0
1     PIN_FLD_AMOUNT      DECIMAL [0] NULL
```

Using testnap to Run Sample Stored Procedure

The following shows how to run the sample **get_gl_total** stored procedure by using the **testnap** utility and the sample input flist:

1. Start the **testnap** utility:

```
testnap
```

2. Load the sample input flist (in a file named **sample_input**) into buffer **1**:

```
r <<token sample_input 1
```

3. Run the PCM_OP_EXEC_SPROC opcode using the input flist from buffer **1**:

```
xop PCM_OP_EXEC_SPROC 0 1
```

   The utility returns the opcode's output flist. For example:

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /procedure -1 0
0 PIN_FLD_RESULTS           ARRAY [0]
1     PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 1536772 0
```

```
1     PIN_FLD_RESOURCE_ID       INT [0] 978
1     PIN_FLD_GL_ID             INT [0] 102
1     PIN_FLD_AMOUNT        DECIMAL [0] 13.548
0 PIN_FLD_RESULTS             ARRAY [1]
1     PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 1536772 0
1     PIN_FLD_RESOURCE_ID       INT [0] 1000502
1     PIN_FLD_GL_ID             INT [0] 102
1     PIN_FLD_AMOUNT        DECIMAL [0] 0
```

12

Using Configuration Files

Learn about the Oracle Communications Billing and Revenue Management (BRM) configuration and properties files, including where to find them and how to edit them.

Topics in this document:

About Configuration Files

The primary purpose of configuration files is to enable the different components of BRM to communicate with each other (see "Connecting BRM Components"). The configuration files can also include other entries that let you increase performance and implement business policies.

- Most BRM components and utilities use configuration files (**pin.conf**).
- BRM Java programs use properties files (usually **Infranet.properties**).

  Any configuration entry that includes an application name, such as **Infranet.pricing_tool.log.file**, is specific to that application. Any other entry is a shared entry, applying to all applications. Any application-specific entry overrides a shared entry.
- BRM programs based on Perl scripts read configuration information from a file named **pin.conf.pl**.

> **Note:**
>
> By default, the BRM Installer stores configuration entries, including the sensitive information (such as database and account passwords), in the Oracle wallet and the BRM applications retrieve the data from the Oracle wallet. For more information, see "About Oracle Wallet".

You can use any text editor to edit configuration files.

> **Note:**
>
> Before you edit a configuration file, save a backup copy.

Some configuration files are write-protected. Before editing the file, remove that restriction. After you edit the file, restore the restriction.

Each configuration file includes specific, detailed information about how to edit each configuration entry in that file. To edit an entry, follow the guidelines provided for that entry.

To insert a comment, type a crosshatch (#) followed by the comment. BRM ignores all text on that line.

Configuration File Locations

The default location for a configuration file is the directory where the system process or program runs. Typical locations are:

- Directories inside the *BRM_home*/**sys** directory (for example, *BRM_home*/**sys/cm/pin.conf**).
- Directories inside the *BRM_home*/**apps** directory (for example, *BRM_home*/**apps/pin_billd/pin.conf**).
- In the application folder (for example, *BRM_home/Application_home*/**Infranet.properties**).

Syntax for Configuration Entries

Entries in a configuration file use the following syntax:

```
host_name     program     keyword     values
```

where:

- *host_name* is the name or IP address of the computer. To refer to the current computer, use a single hyphen (-). If several computers share the same configuration file, use the name of the current computer. In this case, BRM components, such as the Connection Manager (CM) or the Data Manager (DM), use only the entries that contain the host name on which they are started.

- *program* is the name of the program to which this entry applies. The program can be:
  - The name of the application (or custom application) or Facilities Module (FM), such as **cm**, **pin_billd**, or **fm_bill**. Use a specific name when the entry applies only to a single program.
  - **nap** (Network Application Program). Use **nap** when the entry applies to general BRM applications, which use the PCM_CONNECT and PCM_CONNECT_OPEN functions.
  - Blank or a single hyphen (**-**). The entry applies to any BRM function, such as **pin_virtual_time**.
- *keyword* is the name of the configuration entry.
- *values* is one or more parameters specific to the configuration entry. Values are separated by spaces.

A single configuration entry resembles the following example:

```
- cm   userid   0.0.0.1   /service   1
```

This entry applies to the Connection Manager (**cm**) on the local computer (-). The entry is called **userid**, and the three values associated with that entry are **0.0.0.1**, **/service**, and **1**.

> **Note:**
>
> Some configuration files have entries with **userid** and a database number, as shown here. BRM ignores the database number in these entries:
>
> ```
> - cm   userid   0.0.0.1   /service   1
> ```

Syntax for Facilities Module Entries

The CM configuration file includes entries for Facilities Module (FM) that are linked to the CM at startup. Some of these entries are for the base set of FMs that are part of the standard release; other entries are for optional BRM components. You can also add entries for custom FMs.

Configuration entries for FMs use the following syntax:

```
- cm fm_module file_name initialization_table initialization_function tag
```

where:

- *file_name* is the path to the shared library file containing the functions that make up the FM. The file name has the **.so** extension on Oracle Linux and Oracle Solaris.

  Do not change this parameter unless you change the location of the file.
- *initialization_table* is the name of the configuration structure that maps each opcode to a function. Do not change this text for standard FMs.
- *initialization_function* is either a hyphen (**-**), meaning that no function is run when the CM is started, or the name of the function to be run at startup. Some FMs call optional initialization functions that you can use to configure the FM.
- *tag* identifies the FM to the CM. Each CM has an equivalent tag as part of the **cm_ports** configuration entry. Each FM with a matching tag is linked to that CM at startup. The

default tag for the base set of FMs is **pin**. You can use other tags to define separate sets of FMs for multiple CMs on the same computer.

> **Note:**
>
> Some FMs depend on others. Never change the order of the base set of FMs in the CM configuration file.

Guidelines for Editing Java Properties Files

Java applications get configuration information from Java properties files instead of the **pin.conf** files that are used for C applications.

The BRM installation program uses information supplied by the installer to write configuration information to the properties files.

Each properties file includes specific, detailed information about how to edit each configuration entry in that file. To edit an entry, follow the guidelines provided with that entry.

You can add comments to properties files to help others understand the purpose of your entries. To insert a comment, type a crosshatch (#) followed by the comment. BRM ignores all text on the same line after the crosshatch.

Connection entries, failover entries, and other entries each have their own syntax considerations.

Connection Entry

The connection entry consists of a full URL to the BRM services. It takes one of two forms, depending on the type of login setting:

- For the Type 1 login setting, which requires a password, use the following format:

  pcp://*user_name***:***password*@*host_name***:***port***/***service_object*

  where:

  - *user_name* is the login name to use for connecting to BRM.
  - *password* is the password for the specified user name.
  - *host_name* is the name or IP address of the computer running the CM or Connection Manager Master Process (CMMP).
  - *port* is the TCP port number of the CM or CMMP on the host computer. The port number must match the corresponding **cm_ports** entry in the CM or CMMP configuration file.
  - *service_object* is the service type. The trailing number, “1," is the Portal object ID (POID) of the service.

  For example:

  ```
  infranet.connection=pcp://root.0.0.0.1:password@hostname:11960/service/
  admin_client
  ```

- For the Type 0 login setting, which does not require a password, use the following format:

  pcp://*host_name***:***port***/***database_number***/***service_object*

  where:

  - *host_name* is the name or IP address of the computer running the CM or Connection Manager Master Process (CMMP).
  - *port* is the TCP port number of the CM or CMMP on the host computer. The port number must match the corresponding **cm_ports** entry in the CM or CMMP configuration file.
  - *database_number* is the database number assigned to your BRM database when the DM was installed. For example, **0.0.0.1**.
  - *service_object* is the service type. The trailing number, “1," is the Portal object ID (POID) of the service.

  For example:

  ```
  infranet.connection=pcp://hostname:11960/0.0.0.1/service/admin_client
  ```

Failover Entry

A failover entry refers to an alternate CM host that an application can use to connect to BRM if the main host, specified in the connection entry, is unavailable.

Use the following format:

infranet.failover.1=pcp://*host_name*:*port*

where:

- *host_name* is the name or IP address of the computer running the CM or CMMP.
- *port* is the TCP port number of the CM or CMMP on the host computer. The port number must match the corresponding **cm_ports** entry in the CM or CMMP configuration file.

> **Note:**
>
> *user_name*, *password*, and *service_object* for the alternative hosts are the same as for the main host and are not specified in failover entries.

Other Properties Entries

The flags used in the connection entry of the main **Infranet.properties** file are used by all the other properties entries, unless they are overridden.

Other entries that override these values for all your Java applications use the following format:

infranet.*entry*.*specific_entries*

The **Infranet.properties** file also contains entries specific to particular Java applications, in the following format:

infranet.*application*.*specific_entries*

Configuring BRM by Using the pin_bus_params Utility

As part of the BRM installation, a standard group of **/config/business_params** objects is added to the BRM database. These objects contain all the business parameters normally used by BRM. In their default state, these parameters typically disable optional features and direct BRM to use behaviors optimal for most users. However, your business might require optional features or alternative BRM behaviors. Or you might want to add business parameters or parameter classes that are not part of BRM.

If so, use the **pin_bus_params** utility to perform those tasks. This utility has the following capabilities:

- **Retrieving**: Use the utility to retrieve the contents of an existing **/config/business_params** object in your BRM installation and translate it into an XML file that you can edit.
- **Loading**: Use the utility to load the contents of an XML file that contains business parameters into an existing **/config/business_params** object or to create entirely new objects.

You can use the XML file created during retrieval to add new parameters for existing classes or to create an entirely new class of parameters. When you use the utility to load the XML file into the **/config/business_params** object, BRM adds the new parameters to your parameter configuration, and these parameters can be called from custom code. For information on adding new parameters and classes, see *BRM Developer's Guide*.

Retrieving /config/business_params Objects

To retrieve a **/config/business_params** object, run the following command:

```
pin_bus_params -r ParameterClassTag bus_params_ParameterClassName.xml
```

This command retrieves the **/config/business_params** object for the specified class into the specified XML file. Consider the following when retrieving business parameters:

- To retrieve a specific parameter, you must know which parameter class it belongs to. The resulting XML file for each class is short so you can quickly locate specific parameters within the class.
- Because you retrieve one parameter class at a time, you can edit parameters for a single parameter class without affecting any other parameter classes. BRM overwrites only the **/config/business_params** object whose parameter class appears in the resulting XML file, so the overall BRM business parameter configuration remains stable.
- To update more than one parameter class, you must perform multiple retrievals and loads, one for each class.
- You can create a library of class-based business parameter files and individually reload modified versions of these files only when needed.

Loading /config/business_params Objects

To load parameters into **/config/business_params** objects, use the following command:

```
pin_bus_params bus_params_ParameterClassName.xml
```

This command finds the **/config/business_params** object for the parameter class in the XML file and overwrites the object with the new parameters.

Configuring a Shared pin.conf File

If you run several BRM applications and processes on one computer, they can share a single configuration file. To set up a shared configuration file:

1. Combine configuration entries for each BRM component into a single **pin.conf** file.
2. Save that file to the *BRM_home* directory.
3. For each BRM component that uses the shared configuration file, move its specific configuration file to a backup location or rename the file.

When BRM starts any BRM application, component, or process, it searches for the appropriate **pin.conf** file in the following directories in the order shown:

1. The current directory.
2. The system **/etc** directory.

> **Note:**
>
> The **/etc** directory is included in the search path for backward compatibility.

3. If the PIN_HOME environment variable is defined, the *BRM_home***/config** directory.

> **Note:**
>
> If the PIN_HOME environment variable is not defined, BRM skips this part of the search.

Preparing for Platform Migration by Using Variables in pin.conf Files

You can reference certain system environment variables in **pin.conf** configuration files. These references can facilitate future migration of the **pin.conf** files to BRM implementations on other platforms.

For information about environment variables, see *BRM Installation Guide*.

Table 12-1 shows the environment variables that can be referenced in BRM configuration files (**pin.conf**).

Table 12-1 BRM Configuration File Environment Variables

Environment Variable	Reference in pin.conf Files
PIN_HOME	${PIN_HOME}
PIN_LOG_DIR	${PIN_LOG_DIR}
LIBRARYEXTENSION	${LIBRARYEXTENSION}
LIBRARYPREFIX	${LIBRARYPREFIX}

Sample **pin.conf** file with environment variable references:

```
- - pin_virtual_time ${PIN_HOME}/lib/pin_virtual_time_file
- fm_rate tax_supplier_map ${PIN_HOME}/sys/cm/tax_supplier.map
- cm fm_module ${PIN_HOME}/lib/fm_utils/$
{LIBRARYEXTENSION} fm_utils_config fm_utils_init pin
- cm fm_module ${PIN_HOME}/lib/fm_delivery/$
{LIBRARYEXTENSION} fm_delivery_config - pin
```

About Oracle Wallet

By default, the BRM installer stores configuration entries, including sensitive information (such as TLS certificates and database and account passwords), in the Oracle wallet. BRM applications retrieve the data from the Oracle wallet unless the configuration entries are also stored in the **Infranet.properties** and **pin.conf** configuration files. The entries in the configuration files take precedence.

To view the list of configuration entries in the Oracle wallet, see "Viewing Configuration Entries in the Client Wallet".

You can store and retrieve configuration entries from the Oracle wallet by using the **pin_crypt_app** and **pin_config_editor** utilities. You can use these utilities to update the configuration entries in the Oracle wallet; for example, to change connection details, log level, or a password.

> **Note:**
>
> If you change the database password in the Oracle wallet, ensure that you change the database password in all the other configuration entries in the Oracle wallet.

For more information, see:

- Storing Configuration Entries for Java PCM Applications in Client Wallet
- Retrieving Configuration Entries from Client Wallet for Java PCM Applications
- Storing Configuration Entries for BRM PCM Applications in Client Wallet
- Retrieving Configuration Entries from Client Wallet for BRM PCM Applications

Viewing Configuration Entries in the Client Wallet

To view entries in the client wallet:

1. Go to the BRM_*home*/**bin** directory.
2. Run the following command:

```
orapki wallet display -wallet client
```

A list of configuration entries that are stored in the client wallet is displayed.

Storing Configuration Entries for Java PCM Applications in Client Wallet

To store a configuration entry for the Java PCM applications in the client wallet:

1. Go to the *BRM_home*/**bin** directory.
2. Do one of the following:

   On Unix:

   - To store a value, run the following command:

     pin_config_editor -setconf -wallet *clientWalletLocation* **-parameter** *configEntry* **-value** *value*

     where:

     – *clientWalletLocation* is the path to the client wallet.

     – *configEntry* is the configuration entry in the client wallet.

     – *value* is the appropriate value for the respective entry in the client wallet.

     For example, running the following command stores the BRM log level as 1 in the client wallet:

     ```
     pin_config_editor -setconf -wallet opt/brm_wallet -parameter
     infranet.log.level -value 1
     ```

   - To store a password, run the following command:

     pin_config_editor -setconf -wallet *clientWalletLocation* **-parameter** *configEntry* **[-pwd]**

     For example, running the following command stores the password in the client wallet:

     ```
     pin_config_editor -setconf -wallet opt/brm_wallet -parameter
     infranet.cmt.passwd -pwd
     ```

   On Windows:

   - To store a value, run the following command:

     java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.ConfigEditor -setconf -wallet *clientWalletLocation* **-parameter** *configEntry* **-value** *value*

     For example, running the following command stores the BRM log level as 1 in the client wallet:

```
java -cp ".;BC\lib\oraclepki.jar;BC\lib\cet.jar"
com.portal.cet.configEditor -setconf -wallet opt/brm_wallet -parameter
infranet.log.level -value 1
```

- To store a password, run the following command:

  java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.ConfigEditor -setconf -wallet *clientWalletLocation* **-parameter** *configEntry* **[-pwd]**

  For example, running the following command stores the BRM password in the client wallet:

  ```
  java -cp ".;BC\lib\oraclepki.jar;BC\lib\cet.jar"
  com.portal.cet.configEditor -setconf -wallet opt/brm_wallet -parameter
  infranet.cmt.passwd -pwd
  ```

Enter the password for the wallet prompt appears.

3. Enter the client wallet password.

   When you run the utility with the **–pwd** parameter, the Enter the password prompt appears.

4. Enter the password to be stored.

> **Note:**
>
> If you want to store the encrypted password, enter the encrypted password at the prompt.

The password is stored in the client wallet.

Retrieving Configuration Entries from Client Wallet for Java PCM Applications

To retrieve a configuration entry from the client wallet for Java PCM applications:

1. Go to the *BRM_home***/bin** directory.
2. Do one of the following:

   On Unix:

   - To retrieve a value, run the following command:

     pin_config_editor -getconf -wallet *clientWalletLocation* **-parameter** *configEntry*

     For example, running the following command retrieves the value stored for the **infranet.connection** entry:

     ```
     pin_config_editor -getconf -wallet -wallet opt/brm_wallet -parameter
     infranet.connection
     ```

   On Windows:

- To retrieve a value, run the following command:

  java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.ConfigEditor -getconf -wallet *clientWalletLocation* **-parameter** *configEntry*

  For example, running the following command retrieves value stored for the **infranet.connection** entry:

  ```
  java -cp ".;BC\lib\oraclepki.jar;BC\lib\cet.jar" com.portal.cet.ConfigEditor -getconf -wallet  opt/brm_wallet -parameter infranet.connection
  ```

The value of the configuration entry is displayed.

Storing Configuration Entries for BRM PCM Applications in Client Wallet

For BRM PCM applications, you typically store the database user ID, database password, and encryption algorithm in the client wallet by using the **pin_crypt_app** utility. For information about the utility's syntax and parameters, see "pin_crypt_app" in *BRM Developer's Guide*.

To store a configuration entry and value for BRM PCM applications in your client wallet, go to the *BRM_home***/bin** directory and do the following:

1. On Unix, run the following command:

   pin_crypt_app -setconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry* **-value** *value*

   where:

   - *clientWalletLocation* is the full path to the client wallet.
   - *program* is the BRM component to which the configuration entry belongs. For example, enter **dm** if the configuration entry is in the DM **pin.conf** file, enter **cm** if the configuration entry is in the CM **pin.conf** file, and so on.
   - *configEntry* is the name of configuration entry to store in the client wallet, such as **sm_id** or **crypt** for the DM **pin.conf** file.
   - *value* is the value for *configEntry*.

   For example, to store the DM **pin.conf** file entry "**- dm sm_id pin01**" in the client wallet:

   ```
   pin_crypt_app -setconf -wallet $PIN_HOME/opt/brm_wallet -program dm -parameter sm_id -value pin01
   ```

2. On Windows, run the following command:

   java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.PinCrypt -setconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry* **-value** *value*

   For example, to store the pin_billd **pin.conf** file entry "**- nap login_name myName**" in the client wallet:

   ```
   java -cp ".;BC\lib\oraclepki.jar;BC\lib\cet.jar" com.portal.cet.PinCrypt -setconf -wallet c:\pin\wallet\client -program pin_billd -parameter login_name -value myName
   ```

3. At the **Enter the wallet password** prompt, enter your client wallet password.

The utility parameter creates a new entry in your client wallet if one does not exist. If an entry already exists, it overwrites the value.

To store a password for BRM PCM applications in your client wallet, go to the *BRM_home*/**bin** directory and do the following:

1. On Unix, run the following command:

   pin_crypt_app -setconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry* **-pwd**

   For example, to store the DM **pin.conf** file entry "**- dm sm_pw** *password*" in the client wallet:

   ```
   pin_crypt_app -setconf -wallet $PIN_HOME/opt/brm_wallet -program dm -parameter sm_pw -pwd
   ```

2. On Windows, run the following command:

   java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.PinCrypt -setconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry* **-pwd**

   For example, to store the pin_billd **pin.conf** file entry "**- nap login_pw myName**" in the client wallet:

   ```
   java -cp ".;oraclepkijarLocation;cetjarLocation com.portal.cet.PinCrypt -setconf -wallet c:\pin\wallet\client -program pin_billd -parameter login_pw -pwd
   ```

3. At the **Enter the wallet password** prompt, enter your client wallet password.
4. At the **Enter value to be stored in the wallet** prompt, enter the password to store in the wallet.

> **Note:**
>
> If you want to store an encrypted password, enter the encrypted password at the prompt.

The password is stored in the client wallet.

Retrieving Configuration Entries from Client Wallet for BRM PCM Applications

To retrieve a configuration entry from the client wallet for BRM PCM applications:

1. Go to the *BRM_home*/**bin** directory.
2. On Unix, run this command:

   pin_crypt_app -getconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry*

   where:

   - *clientWalletLocation* is the full path to the client wallet.

- *program* is the BRM component to which the configuration entry belongs. For example, enter **dm** if the configuration entry is in the DM **pin.conf** file, enter **cm** if the configuration entry is in the CM **pin.conf** file, and so on.
- *configEntry* is the name of the configuration entry to retrieve from the client wallet, such as **sm_id** or **crypt** for the DM **pin.conf** file.

For example, to retrieve the **sm_id** value for the DM **pin.conf** file:

```
pin_crypt_app -getconf -wallet $PIN_HOME/opt/brm_wallet -program dm -parameter
sm_id
```

3. On Windows, run the following command:

   java -cp ".;oraclepkijarLocation;cetjarLocation" com.portal.cet.PinCrypt -getconf -wallet *clientWalletLocation* **-program** *program* **-parameter** *configEntry*

   For example, to retrieve the **sm_pw** value for the DM **pin.conf** file:

   ```
   java -cp ".;BC\lib\oraclepki.jar;BC\lib\cet.jar" com.portal.cet.PinCrypt -getconf -
   wallet c:\pin\wallet\client -program dm -parameter sm_pw
   ```

4. At the **Enter the wallet password** prompt, enter your client wallet password.

The value of the configuration entry is displayed.

13

Connecting BRM Components

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) components to communicate with each other by using parameters in **pin.conf** configuration files.

Topics in this document:

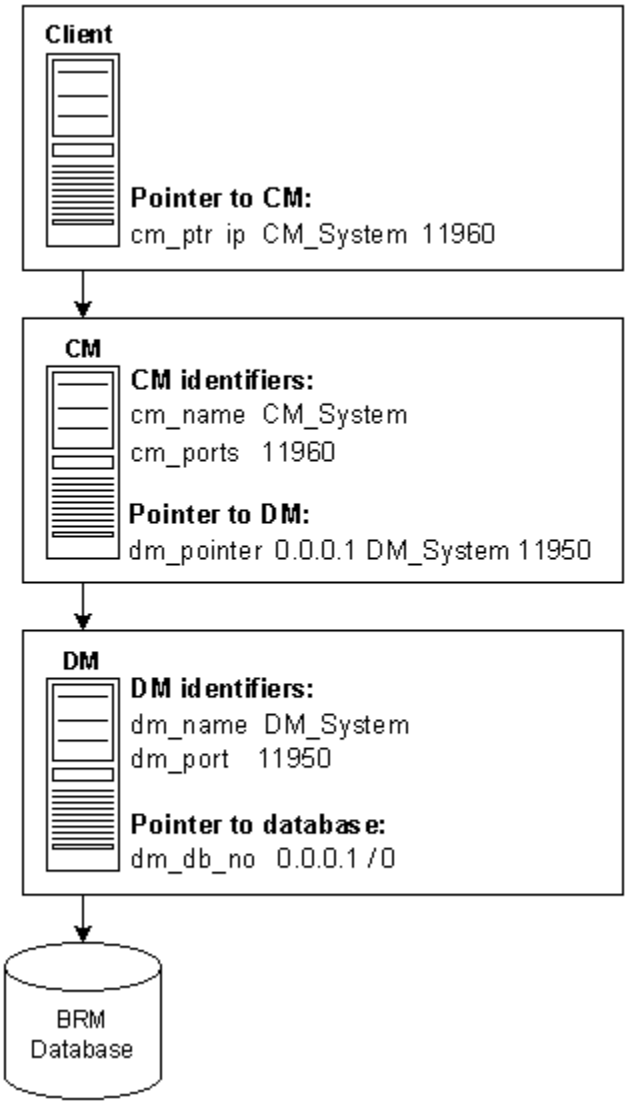
About Connecting BRM Components

To allow BRM components to communicate with each other, you use entries in configuration or properties files. The basic connection entries in the files identify the host names and port numbers of each component.

These connection entries are set when you install BRM and when you install each client application. You can change them if you change your configuration. Depending on how you install BRM, you might have to change some entries to connect BRM components.

Figure 13-1 shows how entries in configuration files link components.

Figure 13-1 BRM Component Connections



In Figure 13-1, the client application is a utility that uses a configuration file entry to point to the CM. (A Java client has a similar entry in its properties file.) The following sample entry includes the CM host name, **CM_System**, and port number, **11960**:

```
cm_ports    ip    CM_System    11960
```

The CM configuration file has corresponding entries:

- The **cm_name** entry identifies the host name as **CM_System**

  ```
  - cm    cm_name    CM_System
  ```

- The **cm_ports** entry identifies the port number as **11960**:

  ```
  - cm    cm_ports    11960    pin
  ```

The CM configuration file includes an entry that points to the DM. This entry includes the DM host name and port number:

```
- cm    dm_pointer    0.0.0.1    ip    DM_System    11950
```

The DM configuration file includes corresponding information:

- The **dm_name** entry identifies the host name as **DM_System**. This entry is optional: if you remove or disable it, BRM uses **gethostname** to find the IP address of the DM computer:

  ```
  - dm dm_name DM_System
  ```

- The **dm_port** entry identifies the port number as **11950**:

  ```
  - dm dm_port 11950
  ```

The DM configuration file specifies the database number, which is in **0.0.*n*.*n* / 0** format:

```
- dm dm_db_no 0.0.0.1 / 0
```

Queue Manager (QM) components (for example, LDAP Manager) use the same types of configuration entries, but they have different names. For example, instead of **dm_max_fe**, the entry is named **qm_max_fe**.

The QM configuration files include the following entry:

- The **qm_port** entry identifies the host address and the port number:

  ```
  - qm_process qm_port host_name port [tag_number]
  ```

  where:

  - *qm_process* is the system process such as **dm_ldap**, **dm_email**, **dm_fusa**, **dm_vertex**, **cm_proxy**, and **dm_invoice**.
  - *host_name* is the host name or the IP address where the system process is deployed.
  - *port* is the port number.
  - *tag_number* is a sequential number that identifies the *host_name* and *port* pair. You use this parameter when configuring multiple **qm_port** entries.

There are additional entries in configuration files, but these entries are the most basic. For more information on connection entries in configuration files, see the comments in the configuration files.

Guidelines for Database and Port-Number Entries

The configuration and properties files specify identifying numbers for databases and DMs. The default numbers, listed in Table 13-1, are systematic to make numbers relatively easy to maintain and extend.

The DM numbers are in the form *A*.*B*.*C*.*D*. Make the number assignments meaningful as follows:

- Use *A* to separate divisions of your company (use **0** if you have none).
- Use *B* to distinguish different BRM installations (use **0** if you have only one).
- Use *C* to indicate the type of DM. For example:
  - **0** for data
  - **1** for transaction processing of credit or debit card
  - **2** for email
  - **3** for taxation

> **Note:**
>
> Start numbering your custom DMs at 100, such as **0.0.100.1**.

- Use *D* to indicate the instance of a particular DM division, installation, or type.

Table 13-1 Database and Port Number Entries

Program	Database Number	Port Number
Oracle DM (**dm_oracle**)	**0.0.0.1**	**12950**
Second Oracle DM for a multischema system (**dm_oracle**)	**0.0.0.2**	**12951**
Each additional **dm_oracle** for multischema systems	**0.0.0.***n*	**12950** + (*n* **– 1**)
Paymentech data manager for credit card, **dm_fusa**	**0.0.1.1**	**12810**
Paymentech data manager for direct debit, **dm_fusa**	**0.0.1.2**	**12810**
Each additional **dm_fusa**	**0.0.1.***n* (*n* > 2)	**12813**+
Email notification manager: **dm_email**	**0.0.2.1**	**12821**
Each additional **dm_email**	**0.0.2.***n*	**12820** + *n*
Activity Log Data Manager of a BRM satellite: **dm_opstore**	**0.0.4.1**	**12841**
Each additional **dm_opstore**	**0.0.4.***n*	**12840** + *n*
LDAP: **dm_ldap**	**0.0.5.1**	**12851**
Each additional **dm_ldap**	**0.0.5.***n*	**12850** + *n*
Wireless Provisioning DM, **dm_prov_telco**	**0.0.10.2**	**10970**
Connection Manager (CM) or Connection Manager Master Process (CMMP)	NA	**11960**
Each additional CM or CMMP	NA	**11960** + *n*
Credit-card or online-processing batch simulators (answer_b)	NA	**5678**
Credit-card or online-processing online simulators (answer_s)	NA	**5679**

Setting Data Manager Attributes

In the CM configuration file, you can set DM attributes (**dm_attributes**) for each DM to which you connect.

Connecting a Data Manager to the BRM Database

Use the following DM **pin.conf** file entries in Table 13-2 to connect a DM to the BRM database. These entries are used by multiple DMs, such as the Oracle DM.

Table 13-2 DM pin.conf File Entries

Entry	Description
sm_database	Specifies the database alias name. For example, for Oracle Database this is the SQL*NET alias defined in the **tnsnames.ora** file. This entry was configured when BRM was installed, so you do not have to change it. **Note:** If you have multiple database hosts, such as an Oracle Parallel Server configuration, include a separate **sm_database** configuration entry for each host.
sm_id	Specifies the database user name that the DM uses to log in to the BRM database. This entry was configured when BRM was installed, but you can change it.

Connecting BRM Utilities

Some BRM utilities, such as **load_tax_supplier**, require you to create a configuration file to tell the utility how to connect to the BRM system. The configuration file must be in the same directory as the utility executable file.

To create a configuration file for a utility:

1. Copy the sample configuration file in *BRM_home*/**source/apps/sample**.

   Use this file, which contains all of the configuration entries needed for connecting to BRM, as a template for any utility configuration file.
2. Edit the configuration entries to reflect your BRM environment. Follow the guidelines in the configuration file.
3. Save the file as **pin.conf** in the directory with the utility executable file.

Table 13-3 shows the common utility **pin.conf** entries:

Table 13-3 Common Utility pin.conf Entries

Entry	Description
cm_ports	Specifies a pointer to the CM or CMMP. Use a separate entry for each CM or CMMP. Each entry includes three values: • *protocol*: **ip** • *host_name*: the name or IP address of the computer running the CM or CMMP • *port*: the port number of the CM or CMMP on this computer The port number should match a corresponding **cm_ports** entry with the same port number in the CM or CMMP configuration file. The default, **11960**, is a commonly specified port for the CM or CMMP. See "About Connecting BRM Components".
login_name	Specifies the login name to use when connecting to the CM.
login_pw	Specifies the password to use when connecting to the CM.
login_type	Specifies if the application requires a login name and password to connect to BRM.

Table 13-3 (Cont.) Common Utility pin.conf Entries

Entry	Description
userid	Specifies the database number and service type for the BRM database. The CM uses the database number to identify the BRM database and to connect to the correct DM. For connections that do not require a login name and password, the CM also passes the service type to the database. The database number, in the form 0.0.0.*n*, is the number assigned to your BRM database when the system is installed. The default is **0.0.0.1**.

In addition, the **pin.conf** entries in Table 13-4 are used by multithreaded application (MTA) utilities:

Table 13-4 MTA Utilities pin.conf Entries

Entry	Description
children	Specifies the number of worker threads spawned to perform the specified work. The default is **5**. **Important:** This entry is mandatory. For more information, see the following sections: • Tuning the Number of Children for Billing Utilities • Setting the Number of Children for Invoice Utilities • Controlling Thread Load on Multithreaded Applications
fetch_size	Specifies the number of records received from the database in a batch and cached in system memory for processing. The default is **5000**. **Important:** This entry is mandatory. See the following sections: • Tuning the Account Cache Size for Billing Utilities (fetch_size) • Tuning the Account Cache Size for Invoice Utilities (fetch_size)
hotlist	Specifies the name for the hotlist file. This parameter is available for backward compatibility.
logfile	Specifies the file name used to log errors. **Important:** This entry is mandatory. See "Changing the Name or Location of a Log File".
loglevel	Error reporting level. • **0**: no logging • **1**: (Default) log error messages only • **2**: log error messages and warnings • **3**: log error, warning, and debugging messages See "Setting the Reporting Level for Logging Messages".
loop_forever	Specifies whether the application goes into an infinite loop: • **0** specifies to not go into an infinite loop. • **1** specifies to go into an infinite loop. The default is **0**.
max_errs	Specifies the maximum number of errors allowed in the application. The application stops when the number of errors exceeds this number. The default is **1**.

Table 13-4 (Cont.) MTA Utilities pin.conf Entries

Entry	Description
max_time	Specifies the maximum time, measured from application start time, for job processing before the application exits. The default is **0**, for infinite time.
monitor	Specifies the file used by the **pin_mta_monitor** utility. The default is **monitor**. **Important:** The file specified is for system use only and should not be deleted or modified.
multi_db	A flag that determines whether the application works with a BRM multischema system. The default is **0**. For information, see "Using Multithreaded Applications with Multiple Database Schemas" in *BRM Developer's Guide*.
per_batch	Specifies the number of objects processed by each worker thread in batch mode. The default is **500**. **Important:** This entry is mandatory.
per_step	Specifies the number of objects returned by each search step. The default is **1000**. **Important:** This entry is mandatory. See "Setting the Batch Size for Invoice Utilities (per_step)".
pin_virtual_time	Enables **pin_virtual_time** to advance BRM time. See "pin_virtual_time" in *BRM Developer's Guide*.
respawn_threads	Re-spawns worker threads if they exit due to an error. Threads are re-spawned if necessary after every search cycle. The default is **0**, for no re-spawning.
retry_mta_srch	The number of retry attempts for main search execution in case of search error. The default is **0**, for no retry.
return_worker_error	Specifies whether to return an error code when any thread encounters an error: • **0** specifies to *not* return an error code. • **1** specifies to return an error code. The default is **0**.
sleep_interval	Specifies the sleep interval time in seconds. The default is **60**.

14

Controlling Batch Operations

Learn how to use the Batch Controller to launch batch handlers that check or process data for your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

See also "Using Configuration Files".

About the Batch Controller

The BRM Batch Controller lets you specify when to run programs or scripts automatically, either at timed intervals or upon creation of certain files, such as log files.

The Batch Controller configuration file (*BRM_home*/**apps/batch_controller/Infranet.properties**) has entries that tell the Batch Controller when to run the specified batch handlers. These programs or scripts can be triggered at specified times or by specified kinds of occurrences, such as creation of a new log file. You can specify *scheduled execution*, starting the handler at fixed times of the day only, or *metronomic execution*, starting the handler repeatedly at a fixed interval.

A batch handler can be any executable program or script that can be run from a command line. For more information about the requirements for writing batch handlers, see "Writing Custom Batch Handlers" in *BRM Developer's Guide*.

Only one Batch Controller can run on a single computer, but it can control many batch handlers to launch various applications.

BRM installation includes a generic batch handler, called SampleHandler, which you can use with most applications. See "About SampleHandler".

Setting Activity Times and Triggers

Certain general parameters apply to all batch activity. Other parameters identify the batch handlers and regulate when those handlers are to be run.

General Batch Controller Parameters

The Batch Controller's Infranet **properties** file (*BRM_home*) specifies how to connect the Batch Controller to the *Connection Manager (CM)*, when to allow high batch traffic, how much logging to record, and how long to wait for handlers.

Connection Parameters

Batch Controller requires the following parameters to connect to a CM: a user ID and password (specified in the **infranet.connection** parameter), the CM's port number, and the address of the CM's host computer.

> **Note:**
>
> To only use a user ID, set the **infranet.login.type** parameter to 0; to use both a user ID and a password, set **infranet.login.type** to 1.

In the **batch.lock.socket.addr** parameter, specify the socket number to lock on. If in doubt, check with your system administrator for the number of an unused socket that can be used exclusively by the Batch Controller. You can use the default value, 11971, unless some other application is using that socket number.

To write a stack trace of any runtime exceptions in the log file, set **infranet.log.logallebuf** to **True**. To disable this feature, set it to **False**.

Time-to-Run Parameters

When your system's heaviest load typically occurs, you might want some handlers to run less often than they do when the load is lighter. The Batch Controller divides the day into high-load and low-load periods to balance the demand for system resources.

Specify the starting time of your system's busiest period in the **batch.start.highload.time** parameter. Specify the starting time of your system's slowest period in the **batch.end.highload.time** parameter. For each of these times, specify the hour, minute, and second, in *hhmmss*, using the 24-hour clock. For each handler, you can specify the maximum number of simultaneous actions during each of these two periods.

The **end** parameter must be greater than the **start** parameter. If you do specify **start**, it must be greater than **end**. Specifying the same value for both parameters causes an error.

In the **batch.check.interval** parameter, specify the time, in seconds, between checks for occurrences of the type specified in **batch.timed.events** or **batch.random.events**. Omitting **batch.check.interval** causes an error.

Log-File Parameter

For logging, you can specify the level of information reported, the full path of the log file, and whether to print a stack trace of runtime exceptions. Set **infranet.log.level** to **0** for no logging; to **1** for error messages only; to **2** for error messages and warnings; or to **3** for error messages, warnings, and debug messages. Set **infranet.log.file** to the path and file name for the Batch Controller log file.

If you omit **infranet.log.file**, BRM uses **default.pinlog** in the current directory. Omitting **infranet.log.level** causes an error.

Timeout-Limit Parameters

You can also set timeout limits for handlers to start their objects and to complete their execution. Set **batch.handler.start.wait** to the number of seconds allowed for the handler to update its own object status from **STARTING** to **STARTED**, and set **batch.handler.end.wait** to the number of seconds allowed for updating from **STARTED** to some other state, such as **COMPLETED**. See "Writing Custom Batch Handlers" in *BRM Developer's Guide* for descriptions of all of the handler states.

> **Note:**
>
> Omitting either **batch.handler.start.wait** or **batch.handler.end.wait** causes an error.

Example of Parameters

This example demonstrates the general parameters:

```
infranet.connection         pcp://root.0.0.0.1:password@myserver:11960/service/
pcm_client
infranet.login.type         1
infranet.log.logallebuf     true
batch.lock.socket.addr      11971
batch.start.highload.time   083000
batch.end.highload.time     211500
infranet.log.file           BRM_home/apps/batch_controller/batch.pinlog
infranet.log.level          2
batch.handler.start.wait    600
batch.handler.end.wait      43200
```

In this example, the Batch Controller logs on to the CM host on **myserver**, port 11960, as user **root.0.0.0.1**. High usage is expected between 8:30 a.m. and 9:15 p.m. Logging writes a normal level of information to file **batch.pinlog** and prints a stack trace if any program errors are found. Timeouts for updating object status are 10 minutes (600 seconds) for going to **STARTED** and 12 hours (43,200 seconds) for going to **COMPLETED** or some other state.

Handler Identification

To identify each of the batch handlers:

1. In the *handler_name***.name** parameter, enter a descriptive label for the handler. This name can include spaces or any other characters. It can be of any length, but short names are easier to read.
2. In the *handler_name***.max.at.highload.time** parameter, specify the highest number of instances of this batch handler that are permitted to run simultaneously during the time from **batch.start.highload.time** to **batch.end.highload.time**.
3. In the *handler_name***.max.at.lowload.time** parameter, specify the highest number of instances of this batch handler that are permitted to run simultaneously during the low-usage time; the time from **batch.end.highload.time** to **batch.start.highload.time**.
4. In the *handler_name***.start.string** parameter, specify the command line to start this batch handler.

> **Note:**
>
> When the Batch Controller issues this command, it appends **-p** *handler_poid* **-d** *failed_handler_poid* to the command, as described in "Writing Custom Batch Handlers" in *BRM Developer's Guide*. If you are not using custom batch handlers, you can ignore these options.

This example demonstrates the identification parameters:

```
handler1.name                     Visa Handler #1
. . .
handler1.max.at.lowload.time      4
handler1.max.at.highload.time     2
handler1.start.string    BRM_home/apps/visa-handler/visa.pl

handler2.name                     Discover Handler #1
. . .
handler2.max.at.lowload.time      5
handler2.max.at.highload.time     3
handler2.start.string      BRM_home/apps/discover-handler/discover -y 2001
```

In this example, the internal name **Visa Handler #1** applies to the program started by the command string *BRM_home***/apps/visa-handler/visa.pl**, which runs a Perl script. The parameters in the above example limit this program to one or two concurrent actions during the specified high-load period or as many as four during the low-load period.

The other batch handler in this example, **Discover Handler #1**, runs the application

*BRM_home***/apps/discover-handler/discover** with the additional option **-y 2001**.

Occurrence-Driven Execution

To trigger execution based on specified occurrences:

1. In the **batch.random.events** parameter, specify the event identifiers. If you have two or more event identifiers, separate each with a comma, but no blank space.
2. In the *event_name***.name** parameter, enter a descriptive label for the event. This name can include spaces or any other characters. It can be of any length, but short names are easier to read.
3. In the *event_name***.handlers** parameter, specify the identifiers of one or more handlers to trigger when the event occurs. You must specify at least one handler. If you have two or more handlers, separate each with a comma; no blank spaces are allowed.
4. In the *event_name***.file.location** parameter, specify the full path name of the directory to monitor for the arrival of new files that match the pattern in *event_name***.file.pattern**.
5. In the *event_name***.file.pattern** parameter, specify the file name pattern to look for. You can use an asterisk (*) to represent zero or more characters in the file name. No other wild cards (metacharacters) are supported.

> **Note:**
>
> - Depending on your configuration, the file pattern might conflict with a file pattern used by another component, such as Rated Event Loader. To prevent conflicts, use a specific pattern, for example, **test*.out**.
> - You must specify **batch.timed.events** or **batch.random.events** or both. Specifying neither causes the Batch Controller to shut down just after it starts because it has no tasks to perform.

When the Batch Controller starts, it tests the file name pattern against every file name in the specified directory and runs the batch handler for each file where the name matches the pattern. It then monitors the files entering that directory and runs the batch handler whenever it finds a match.

> **Note:**
>
> By default, the Batch Controller appends **.bc** to the file name of each file it processes. This prevents batch handlers from retrieving files that the Batch Controller has yet to process. You can change the default **.bc** extension by editing the **batch.file.rename.extension** entry in the Batch Controller **Infranet.properties** file.

Timed Execution

The Batch Controller provides two time-based scheduling options: metronomic execution and scheduled execution.

Metronomic Execution

To set up metronomic execution:

1. In the **batch.timed.events** parameter, specify the event identifiers. If you have two or more event identifiers, separate each with a comma; blank spaces are not allowed.
2. In the *event_name***.name** parameter, enter a descriptive label for the event. This name can include spaces or any other characters. It can be of any length, but short names are easier to read.
3. In the *event_name***.handlers** parameter, specify identifiers for one or more handlers to trigger when the event occurs. If you have two or more handlers, separate each with a comma; blank spaces are not allowed.
4. (Optional) In the *event_name***.start** parameter, specify when you want the first execution to occur, in *hhmmss*, using the 24-hour clock. If you omit this parameter, the first execution occurs immediately after the Batch Controller starts.
5. In the *event_name***.interval** parameter, specify the frequency, in seconds, for triggering the associated handler. Failing to specify the interval causes an error.
6. (Optional) In the *event_name***.count** parameter, specify how many times to run this batch handler. If you do not specify this limit, batch handlers run repeatedly at the fixed interval for as long as the Batch Controller is running.

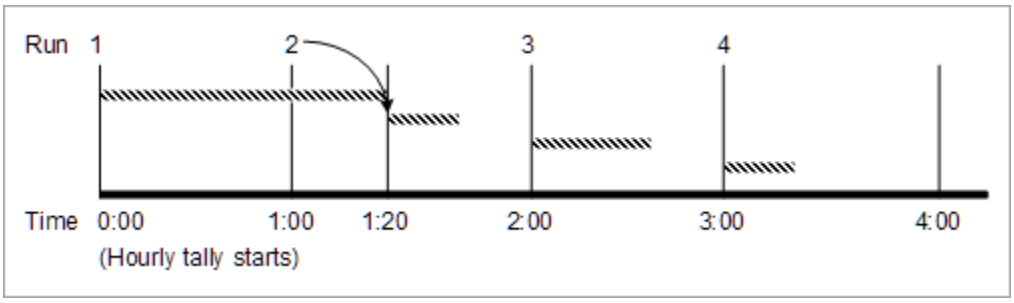
> **Note:**
>
> You must specify **batch.timed.events** or **batch.random.events** or both. Specifying neither causes the Batch Controller to shut down just after it starts because it has no tasks to perform.

This example demonstrates the metronomic parameters:

```
batch.timed.events                event4
. . .
event4.name                       Hourly tally
event4.handlers                   handler4
event4.start                      000000
event4.interval                   3600
event4.count                      4
```

In this example, the occurrence specified as **event4** is named **Hourly tally**. It runs **handler4** for the first time at midnight (**000000**), and then runs it again every hour (**3600** seconds) after that until it has run four times. If it cannot run at a scheduled time because previous executions are not finished, it runs again immediately as soon as possible. For example, consider the time line in Figure 14-1 for **event4**, above:

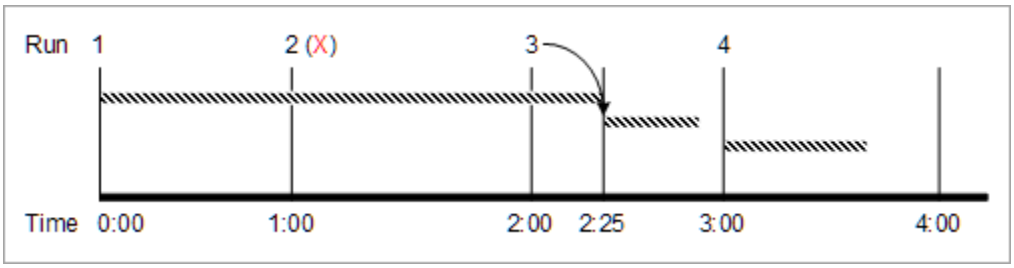
Figure 14-1 Hourly Tally Run 1 Exceeds 1 Hour



In this example, the first run of **handler4** continues for more than an hour, taking it past the time when the second run is supposed to begin. The second scheduled run cannot start at the one-hour interval, so it starts as soon as possible after that (1:20 a.m.). The third and fourth scheduled executions start at regular multiples of the interval, measured from the first run.

If the overly long run continues past two scheduled run start times (occurrences), only one run starts on the delayed basis. For example, suppose the midnight run lasts until 2:25 a.m., as shown in Figure 14-2:

Figure 14-2 Hourly Tally Run1 Exceeds 2 Hours



In this case, the run scheduled for 2:00 begins immediately at 2:25. The fourth run, scheduled for 3:00 begins on time. The second run, scheduled for 1:00 is skipped.

Scheduled Execution

To set up scheduled execution:

1. In the **batch.timed.events** parameter, specify the event identifiers. If you have two or more event identifiers, separate each with a comma; blank spaces are not allowed.
2. In the *event_name***.name** parameter, enter a descriptive label for the event. This name can include spaces or any other characters. It can be of any length, but short names are easier to read.
3. In the *event_name***.handlers** parameter, specify identifiers for one or more handlers that are to be triggered when the event occurs. If you have two or more handlers, separate each with a comma; blank spaces are not allowed.
4. In the *event_name***.at** parameter, specify each time when you want execution to occur, in *hhmmss*, using a 24-hour clock.
5. In the *event_name***.file.location** parameter, specify the full path name of the directory to monitor for the arrival of new files that match the pattern in *event_name***.file.pattern**.
6. In the *event_name***.file.pattern** parameter, specify the file name pattern to look for. You can use an asterisk (*) to represent zero or more characters in the file name. No other wild cards (metacharacters) are supported.

> **Note:**
>
> Depending on your configuration, the file pattern might conflict with a file pattern used by another component, such as Rated Event Loader. To prevent conflicts, use a specific pattern, for example, **test*.out**.

Starting the Batch Controller

Use this command to start the Batch Controller:

```
start_batch_controller
```

About SampleHandler

BRM software includes a generic batch handler, called SampleHandler, which you can use with any batch application that processes data from files. The Batch Controller can call this batch handler whenever a specified directory receives a file whose name matches a specified pattern. The input and output files are then moved to directories that you specify.

Preparing SampleHandler for use involves:

1. Copying SampleHandler
2. Customizing SampleHandler
3. Configuring the Batch Controller
4. Starting the New Batch Handler

If SampleHandler does not meet your needs, you can write your own batch handler, as described in "Writing Custom Batch Handlers" in *BRM Developer's Guide*.

Copying SampleHandler

The directory *BRM_home***/apps/sample_handler** contains these files:

- **pin.conf:** the batch handler's configuration file for BRM-related parameters.
- **SampleHandler.pl**: the actual code of the batch handler.
- **SampleHandler_config.values:** the batch handler's configuration file for application-specific parameters.

Copy the entire directory and name the copy appropriately. For example, if your new handler is for the Widget application, then you might name the new directory *BRM_home***/apps/**<*widget_handler*>.

In the new directory, you can rename the **SampleHandler_config.values** file to include the application's name, such as <*widget_handler*>**_config.values**. If you do so, you must also edit the **SampleHandler.pl** file to change **SampleHandler_config.values** to the new name.

Customizing SampleHandler

To configure the new batch handler for the desired application:

1. Open the **pin.conf** file for the batch handler (*BRM_home***/apps/**<*widget_handler*>**/ pin.conf**, for example).
2. Edit the BRM connection parameters. For information, see "Using Configuration Files".
3. Save and close the batch handler's **pin.conf** file.
4. Open the **.values** file for the batch handler (*BRM_home***/apps/**<*widget_handler*>**/ SampleHandler_config.values**, unless you have renamed the file).
5. Ensure that the **$HANDLER_DIR** entry specifies the full path to the directory containing the batch application's log, input, output, and other files.
6. Edit the **$FILETYPE** entry to specify the file name pattern to look for.

The batch handler retrieves all files with this file name pattern from the specified directory.

> **Note:**
>
> - The file name pattern must have the **.bc** file extension. The Batch Controller automatically appends **.bc** to each file name before it runs a batch handler.
> - You can use an asterisk (*****) to represent zero or more characters in the file name. No other wild cards (metacharacters) are supported.

7. (Optional) To change the batch handler's log file to a directory other than **$HANDLER_DIR**, edit the **$LOGFILE** entry to specify the full path to the desired directory.
8. Edit the **$pinUEL** entry to specify the name of the application to run.

   Ensure that the **$pinUELDir** entry specifies the full path to the directory containing the application to run.
9. (Optional) To configure the batch handler to get input files from a directory other than **$HANDLER_DIR**, edit the **$STAGING** entry to specify the full path to the desired directory.

   The batch handler will move input files from the **$STAGING** directory to the **$PROCESSING** directory, where the application will read them.
10. (Optional) To configure the application to get input files from a directory other than **$pinUELDir**, edit the **$PROCESSING** entry to specify the full path to the desired directory. This must be the same directory that is specified as the application's input directory.

    The batch handler will move input files from the **$PROCESSING** directory to the **$ARCHIVE** or **$REJECT** directory, depending on the application's exit code. Successfully processed files go into the **$ARCHIVE** directory, and files with problems go into the **$REJECT** directory.
11. (Optional) To store the application's processed files somewhere other than **$HANDLER_DIR**, edit the **$ARCHIVE** and **$REJECT** entries to specify the full paths to the desired directories.
12. Save and close the batch handler's **.values** file.

Configuring the Batch Controller

You must identify your new batch handler to the Batch Controller. Edit the **Infranet.properties** file of the Batch Controller, as described in "Handler Identification".

Starting the New Batch Handler

As with any batch handler, you start this one by starting or restarting the Batch Controller. The Batch Controller monitors the newly specified file location for the arrival of files and, when a file appears, starts the new batch handler. For more information, see "About the Batch Controller".

Before using the new batch handler for your production system, you should try it on a development system.

15

About Connection Pooling

Learn about the connection pool functionality in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Overview

A connection pool is a set of connections maintained between an application, such as Conversion Manager, and the Connection Manager (CM). An incoming request is assigned a connection from this pool and uses the connection to perform operations. When the operation completes, the connection is returned to the pool.

If an incoming request cannot be assigned a connection immediately, the request is queued. The request waits for a connection to become available for a configurable period of time. If a connection doesn't become available during this time, an exception is thrown indicating that the request timed out.

Connection pooling includes these features:

- Automatic connection pool resizing
- Automatic removal and replacement of bad connections
- Automatic connection attempt retries (failover)
- Timeout management

Configuring the Connection Pool

You configure the connection pool by using attribute-value pairs in the application's **Infranet.properties** file.

> **Note:**
>
> For the location of the **Infranet.properties** file, descriptions of any application-specific parameters, and parameter default values, see the appropriate application documentation.

Table 15-1 describes the configurable connection pool parameters in the **Infranet.properties** file.

Table 15-1 Configurable Connection Pool Properties

Parameter	Description
infranet.connectionpool.maxsize	The maximum number of connections the connection pool maintains.
infranet.connectionpool.minsize	The minimum number of connections the connection pool maintains. **Note:** When you first start the connection pool, it may have fewer connections than the *minsize* value. When the *minsize* number of connections is reached, the number of connections will not fall below this count.
infranet.connectionpool.timeout	The time *in milliseconds* that a connection request will wait in the pending request queue for a free connection before it times out. If a pending request doesn't receive a connection during this time, an exception is thrown.
infranet.connectionpool.maxidletime	The time *in milliseconds* that an idle (unused) connection remains in the connection pool before it is removed. **Important:** If the value is set too low, connections might be removed and restored too frequently. This can degrade system performance.

Connection Pool Error Handling

Connection pool handles errors by throwing determinate exceptions. These exceptions, derived from the **DeterminateException** class, specify causes for connection failure as shown in Table 15-2:

Table 15-2 Connection Pool Error Handling

Reason for exception thrown	Description
CONNECT_PARSE_ERROR	Thrown when parsing the property file name-value pairs results in errors.
CONNECT_TIMED_OUT	Thrown when a connection cannot be returned from a pool due to request-timeout.
CONNECTION_COULD_NOT_BE_ESTABLISHED	Thrown when there are no connections available and the pool is expanding in size. The additional connections had problems establishing connections to BRM.
CONNECTION_NO_LOGIN_INFO	Thrown when the connections created cannot log on to BRM.
CREATE_ERROR	Thrown by any **createInstance** APIs if the static instance is already created.
WAIT_ERROR	Thrown when a connection is in the request queue.

Note:

- To find the reason for the exception, use this line in your code:

  ```
  get toString()
  ```

- If an exception is thrown, the calling code is responsible for issuing a new connection request.

Monitoring Connection Pooling Events

When a request is assigned a connection or a connection request times out, the connection pool messages are recorded in the log file.

16

Running Utilities in Multischema Systems

Learn how to run utilities in an Oracle Communications Billing and Revenue Management (BRM) multischema system.

Topics in this document:

See also "Managing a Multischema System".

About Utilities that Support Multischema Systems

The utilities listed in Table 16-1 can be configured to support global searches across all database schemas in a multischema system.

Table 16-1 Utilities Supporting Multischema Systems

Utility Type	Utility Name
Billing	**pin_bill_accts** **pin_trial_bill_accts**
Customer Management	**pin_state_change**
Invoicing	**pin_inv_accts** **pin_inv_doc_gen** **pin_inv_export**
Payment	**pin_collect** **pin_deposit** **pin_refund**
General Ledger	**pin_ledger_report**
Analytics	**pin_generate_analytics**

The way you configure one of these utilities to run against multiple schemas depends on how it is run:

- **Through Business Operations Center**: These utilities are run by Business Operations Center during basic operations such as billing, invoicing, and payment collection. You configure whether these operations are performed against all schemas or against a specific schema by using the Business Operations Center GUI only. See "Specifying Job Frequency Settings" in *Business Operations Center Online Help*.
- **Manually at the command line**: You configure a utility to run against multiple schemas by editing its **pin.conf** file.

If you are running one of these utilities manually, you configure it to run against all schemas by doing the following:

1. Open the utility's **pin.conf** file in a text editor.

   For example, for **pin_ledger_report**, you would open the *BRM_home***/apps/pin_billd/pin.conf** file.

2. Set the **multi_db** entry **1**:

   ```
   - pin_mta multi_db   1
   ```

   > **Caution:**
   >
   > If the utility is being run through Business Operations Center, the **multi_db** entry must be set to **0**.

3. Save and close the file.

About Non-MTA Utilities that Support Multischema Systems

You can run the following non-MTA utilities against all schemas in a multischema system or against a specified schema:

- **pin_clean** (see "pin_clean" in *BRM Configuring and Collecting Payments*)
- **pin_recover** (see "pin_recover" in *BRM Configuring and Collecting Payments*)

By default, these utilities run against all schemas.

To run these utilities against only one schema in a multischema system, include the **-schema** *schema_number* parameter with the utility. For example, if your BRM database contains three schemas, you would run the **pin_clean** utility against only the primary schema by running this command:

```
pin_clean -schema 1
```

About Non-MTA Utilities that Do Not Support Multischema Systems

The following non-MTA utilities can connect to only one CM and its associated schema at a time:

- **pin_mass_refund** (see "pin_mass_refund" in *BRM Managing Accounts Receivable*)
- **pin_config_distribution** (see "pin_config_distribution")

To run these utilities in a multischema system, you must have a CM for each schema in the system. You then connect and run the utility against each CM and schema pair. For example, to run a non-MTA utility on a multischema system:

- Connect the utility to the primary CM and run the utility against the primary schema.
- Connect the utility to a secondary CM and run the utility against its associated secondary schema.

- Connect the utility to another secondary CM and run the utility against its associated secondary schema.

To run a non-MTA utility in a multischema system:

1. Connect the utility to the primary CM:
   a. On the primary CM machine, open the *BRM_home*/**apps/pin_billd/pin.conf** file in a text editor.
   b. Edit the following entries:

```
- nap    cm_ports    ip     PrimaryHost    PrimaryPort
- nap    login_name         PrimaryLoginName
- nap    login_pw           PrimaryLoginPassword
- nap    login_type         LoginType
- -      user_id            0.0.0.x /service/pcm_client 1
```

   c. Save and close the file.
2. On the primary CM machine, go to the *BRM_home*/**apps/pin_billd** directory and run the utility.
3. Connect the utility to a secondary CM:
   a. On the secondary CM machine, go to the *BRM_home*/**apps/pin_billd** directory and open the **pin.conf** file in a text editor.
   b. Edit the following entries:

```
- nap    cm_ports    ip     SecondaryHost    SecondaryPort
- nap    login_name         SecondaryLoginName
- nap    login_pw           SecondaryLoginPassword
- nap    login_type         LoginType
- -      user_id            0.0.0.x /service/pcm_client 1
```

   c. Save and close the file.
4. On the secondary CM machine, go to the *BRM_home*/**apps/pin_billd** directory and run the utility.
5. Repeat steps 3 and 4 for each remaining secondary schema in your system.

17

System Administration Utilities and Scripts

Learn about the syntax and parameters for the Oracle Communications Billing and Revenue Management (BRM) system administration utilities and scripts.

Topics in this document:

load_pin_event_record_map

Use this utility to load the event record file (*BRM_home***/sys/data/config/pin_event_record_map**) into the BRM database.

The event record file contains a list of event types to exclude from being recorded in the database. For more information, see "Managing Database Usage".

> **Note:**
>
> At the time you load the event record file, if any events of a type specified in the file already exist in the database, these events remain in the database.

After running this utility, you must stop and restart the Connection Manager (CM).

Location

*BRM_home***/bin**

Syntax

```
load_pin_event_record_map [-d] [-v] | [-r] pin_event_record_map
```

Parameters

-d
Logs debugging information.

-v
Displays detailed information as the event record map is created.

-r
Returns a list of the events in the **pin_event_record_map** file configured to not be recorded.

> **Note:**
>
> This option must be used by itself and does not require the file name.

pin_event_record_map
The file containing the event types to exclude from the database.
Specify to not record the event type by setting its flag value to **0** in the file. To temporarily record an event type, change the event's flag value to **1** and reload the file.
The following example shows the event record file format where the event type is followed by its flag value:

```
/event/session : 1
/event/customer/nameinfo : 0
/event/billing/deal/purchase : 0
/event/billing/product/action/purchase : 0
/event/billing/product/action/modify : 0
```

> **Note:**
>
> The record file includes one default event type, **/event/session**, set to be recorded. Never exclude this event type or any event type that is updated more than once during the same session from recording. If such events are configured not to record, an error occurs when the system tries to update the same event during the same session. To eliminate the error, remove the event causing the error from the record map file.

Results

The utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

partition_utils

Use the **partition_utils** utility to do the following:

- Add partitions
- Purge objects without removing partitions

- Remove partitions
- Enable delayed partitions
- Update partitions
- Restart a **partition_utils** job
- Find the maximum POID for a date

For more information, see "Partitioning Tables" and "About Purging Data".

> **Note:**
>
> - Before you use this utility, configure the database connection parameters in the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Configuring a Database Connection".
> - After you start this utility, do not interrupt it. It might take several minutes to complete an operation depending on the size of your database.

You can run the **partition_utils** utility in the following modes:

- Adding Partitions
- Purging Partitions
- Removing Partitions
- Enabling Delayed Partitions
- Updating Partitions
- Restarting a partition_utils Job
- Finding the Maximum POID for a Date

Location

*BRM_home***/bin**

Adding Partitions

Applies the new partition allocations across all of the partitioned tables of the type specified.

> **Note:**
>
> To use this utility to add a partition for a storable class other than event, bill, invoice, item, journal, newsfeed, sepa, or user activity, you must enable partitioning for that storable class. See "Converting Nonpartitioned Classes to Partitioned Classes".

Syntax for Adding Partitions

```
partition_utils    -o add -t realtime|delayed|prepaid -s start_date
                   -u month|week|day -q quantity
                   [-c storable_class] [-w width]
                   [-f] [-p] [-b] [-h]
```

Parameters for Adding Partitions

-o add
Adds partitions.

-t realtime|delayed|prepaid
Adds real-time, delayed-event, or prepaid-event partitions. Only event tables can have delayed-event partitions.

> **Note:**
>
> Conversion Manager does not support the migration of data to the EVENT_T tables.

-s *start_date*
Specifies the starting date for the new partitions. The format is *MMDDYYYY*. *start_date* must be the day after tomorrow or later. You cannot create partitions starting on the current day or the next day. For example, if the current date is January 1, the earliest start date for the new partition is January 3 (for example, 01032024).

-u month|week|day
Specifies the time unit for partitions.

-q *quantity*
Specifies the number of partitions to add.
If a partition with a future date already exists in the table, this adds more partitions than the specified quantity.
For example, if you want to create one partition with a February 1 start date and the table already contains a P_R_02282032 partition, the P_R_02282032 partition is split into two partitions named P_R_02012032 and P_R_02282032.

-c *storable_class*
Specifies the class of objects to be stored in the partition. The default is **/event**.
If you are adding a partition for a storable class other than an event, bill, invoice, item, or journal, you must enable partitioning for that storable class. See "Converting Nonpartitioned Classes to Partitioned Classes".

-w *width*
Specifies the number of units in a partition, such as 3.

-f
Forces the creation of a partition when *start_date* falls within the time period of the current partition. The existing partition is split in two: one new partition containing objects created before the specified start date and the other new partition containing objects created on or after the specified start date.

> **Note:**
>
> - Before forcing partitions, stop all BRM server processes.
> - The **-f** parameter works differently when you remove partitions. In that case, it forces the removal of objects associated with open items.

-p
Writes a SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

-b
Creates backdated partitions.

-h
Displays the syntax and parameters for this utility.

Purging Partitions

Deletes processed events and non-balance impact events from partitions where the complete partition lies between the start date and end date.

Only event, item, bill, invoice, journal, newsfeed, and sepa objects can be purged without removing their partitions. To purge other types of objects, see "Removing Partitions".

Syntax for Purging Objects without Removing Partitions

```
partition_utils  -o purge [-s start_date] -e end_date [-c storable_class] [-t realtime|
delayed|prepaid]
                 [-p] [-h]
```

Parameters for Purging Objects without Removing Partitions

-o purge
Purges event, item, bill, invoice, journal, newsfeed, and sepa objects without removing their partitions. The event objects must be associated with closed items. To enable this utility to purge event objects associated with open items, see "Enabling Open Items to Be Purged".

-s *start_date*
Specifies the start of the date range containing the objects you want to purge. The date is inclusive. The format is *MMDDYYYY*. If *start_date* is not specified, all objects created on or before *end_date* are purged.

-e *end_date*
Specifies the end of the date range containing the objects you want to purge. The date is inclusive. The format is *MMDDYYYY*.

> **Note:**
>
> If the specified start and end dates do not match the partition boundaries, only objects in partitions that are completely within the date range are purged.

-c *storable_class*
Specifies the objects to purge by base storable class. The default is **/event**.

-t realtime|delayed|prepaid
Purges real-time objects, delayed-event objects, or prepaid objects. To purge all object types, use the **-t** parameter without any options. If this parameter is omitted from the syntax, an error occurs.

-p
Writes a SQL statement that shows the partitions that will be removed to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

-h
Displays the syntax and parameters for this utility.

Removing Partitions

Drops the subset of real-time, delayed-event, or prepaid partitions that occur between the specified start date and end date.

> **Note:**
>
> To remove bill, event, invoice, item, journal, newsfeed, and sepa objects that:
>
> - Meet the purging criteria, see "Purging Partitions".
> - Do not meet the purging criteria, use the **-f** parameter to remove the partitions that contain them. See "Purging Objects by Removing Partitions".
>
> For information about purging criteria, see "Objects Purged by Default".

Syntax for Removing Partitions

```
partition_utils  -o remove -s start_date -e end_date [-t realtime|delayed|
prepaid] [-c storable_class]
                 [-f] [-p] [-h]
```

The only way to purge objects other than bill, event, invoice, item, journal, newsfeed, and sepa from your database is to remove their partitions by using the **-f** parameter.

Parameters for Removing Partitions

-o remove
Removes partitions.

-s *start_date*
Specifies the start of the date range for the objects you want to remove. The format is *MMDDYYYY*.
By default, *start_date* must be at least 45 days ago. You can change this limitation by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".

If the specified dates do not match the partition boundaries, only objects in partitions that are completely within the date range are removed. See "About Purging Database Objects".

> **Note:**
>
> The object's oldest partition is kept by default. To force the removal of the oldest partition, you must set *start_date* to **01011970**.

-e *end_date*
Specifies the end of the date range for objects you want to remove. The format is *MMDDYYYY*.
By default, *end_date* must be at least 45 days ago. You can change this limitation by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".

-c *storable_class*
Specifies the partition to remove by base storable class. The default is **/event**.
When you remove a partition, it removes partitions in all partitioned tables for the specified base storable class and its subclasses.

-t realtime|delayed|prepaid
Removes real-time, delayed-event, or prepaid partitions. The default is to remove all partition types.

-f
Forces the removal of partitions whether or not the objects in the partitions satisfy the purging criteria. For information about purging criteria, see "Objects Purged by Default".

> **Caution:**
>
> Operations using the **-f** parameter cannot be undone and will remove objects that are being used. Use with caution.

-p
Writes a SQL statement that shows the partitions that will be removed to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

-h
Displays the syntax and parameters for this utility.

Enabling Delayed Partitions

Enables delayed partitions for the specified event storable class and all of its parent storable classes.

Syntax for Enabling Delayed Partitions

```
partition_utils  -o enable -t delayed -c storable_class [-p] [-h]
```

Parameters for Enabling Delayed Partitions

-o enable -t delayed
Enables delayed-event partitions.

-c *storable_class*
Specifies the event storable class for which you want to add delayed-event partitions. Delayed-event partitions cannot be used for non-event storable classes.
To add delayed-event partitions for all subclasses of an event, use the percent sign (**%**) as a wildcard (for example, **-c /event/session/%**).

-p
Writes a SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

-h
Displays the syntax and parameters for this utility.

Updating Partitions

Ensures that future partitions are synchronized across all partitioned tables of given class. To partition new tables appropriately, update partitions after adding a new subclass or after adding an array or substruct to an existing class.

Syntax for Updating Partitions

```
partition_utils  -o update [-c storable_class] [-p] [-h]
```

Parameters for Updating Partitions

-o update
Aligns partitions across all object tables for a single base storable class. All real-time and delayed-event partitions get the same real-time partitioning scheme as their base table (EVENT_T for event base storable class tables, ITEM_T for item base storable class tables, and so on).

-c *storable_class*
Specifies the class of objects to update. The default is **/event**.

-h
Displays the syntax and parameters for this utility.

Restarting a partition_utils Job

Runs the last operation that was unsuccessful due to an error, or it cleans its status.

Syntax for Restarting a partition_utils Job

```
partition_utils  -o restart [-b] [-h]
```

Parameters for Restarting a partition_utils Job

-o restart
Restarts the previous operation that was unsuccessful due to an error or abnormal termination.

-b
Bypasses running the previous operation and instead cleans the status of it.

-h
Displays the syntax and parameters for this utility.

Finding the Maximum POID for a Date

Returns the maximum POID for the specified date and partition type.

Syntax for Finding the Maximum POID for a Date

```
partition_utils  -o maxpoid -s date -t realtime|delayed|prepaid [-p] [-h]
```

Parameters for Finding the Maximum POID for a Date

-o maxpoid
Returns the maximum POID for the specified date.

-s *date*
Specifies the date for which the maximum POID is to be found. The format is *MMDDYYYY*.

-t realtime|delayed|prepaid
Gets the maximum POID in only real-time partitions, only delayed-event partitions, or only prepaid partitions.

-p
Writes a SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

-h
Displays the syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the **partition_utils.log** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file. The **partition_utils.log** file includes SQL statements if you use the **-p** parameter.

partitioning.pl

Use this script to manage all the nonpartitioning-to-partitioning upgrade tasks. Run it from a UNIX prompt.

For more information, see "Converting Nonpartitioned Classes to Partitioned Classes".

> **Note:**
>
> This script needs the **partition.cfg** configuration file in the directory from which you run the utility.

Location

BRM_home/**bin**

Syntax

```
perl partitioning.pl [-c | -n | -a | -h ]
```

Parameters

-c
Creates the database objects required for the upgrade, including the following:

- The UPG_LOG_T table that logs all the information about the upgrade
- The **pin_upg_common** package that contains all the common routines for the upgrade

-n
Displays the event tables that will be partitioned during the upgrade.
Tables selected for partitioning are listed in the TABLES_TOBE_PARTITIONED_T table, which is created during the upgrade process. This table contains two columns:

- **table_name**: The name of the table to be partitioned.
- **partition_exchanged**: The value of the exchanged partition. This value is used by the upgrade scripts to perform the table partitioning.

Use the **INSERT** statement to partition tables and use **0** for the **partition_exchanged** value. For example, to insert MY_CUSTOM_EVENT_TABLE, run the following SQL statement:

```
INSERT INTO TABLES_TOBE_PARTITIONED_T (table_name, partition_exchanged)
VALUES ('MY_CUSTOM_EVENT_TABLE',0); COMMIT;
```

> **Note:**
>
> To prevent a listed table from being partitioned, use the SQL **DELETE** statement to delete its name from TABLES_TOBE_PARTITIONED_T.

-a
Partitions the tables listed in the TABLES_TOBE_PARTITIONED_T table.
To partition additional tables, see "Converting Nonpartitioned Classes to Partitioned Classes".

-h
Displays the syntax and parameters for this utility.

Results

The utility does not notify you if it was successful. Look in the UPG_LOG_T table to find any errors.

pin_close_items

Use this utility to close open item objects processed in past billing cycles. Run it from a UNIX prompt.

This utility calls the *BRM_home*/**apps/partition_utils/sql_utils/oracle/pin_close_items.plb** stored procedure.

Before you use this utility, configure the database connection parameters in the **partition_utils.values** file in *BRM_home*/**apps/partition_utils**. See "Configuring a Database Connection".

For more information, see "Closing Open Item Objects Processed in Past Billing Cycles".

> **Note:**
>
> This utility needs the **partition_utils.values** configuration file in the directory from which you run the utility.

Location

BRM_home/**apps/partition_utils**

Syntax

```
pin_close_items [-h]
```

Parameters

-h
Displays the syntax and parameters for this utility.

Results

The utility does not notify you if it was successful. Look in the *BRM_home*/**apps/partition_utils/pin_close_items.log** file to find any errors.

pin_ctl

Use this utility to start and stop BRM components.

> **Note:**
>
> To connect to the BRM database and configure the processes, this utility requires a configuration file in the directory from which you run the utility. This configuration file must be called **pin_ctl.conf**, which is different from most BRM configuration file names.

Syntax

```
pin_ctl action component [-c file_name] [-collectdata] [-debug] [-i]
```

Parameters

action
Specifies the type of action to be run. See "Parameters for Actions".

For example, to start the CM, use the following command:

```
pin_ctl start cm
```

component
Specifies the process on which the action is performed. See "Parameters for Components".

-c *file_name*
Specifies a configuration file to use instead of the default. Use this parameter to run different configurations of the same system.

-collectdata
Gets diagnostic data when starting, stopping, or checking the status of a component.

-debug
Displays debugging information.

-i
Allows the utility to ask whether you want to proceed. This is especially useful when running **stop**, **halt**, and **clear**.

Parameters for Actions

clear
Deletes log entries associated with the component (not the file).

> **Note:**
>
> The log file is not deleted, just the entries.

cstart
Clears the component logs and, if the component is not running, starts the component.
If the component is already running, the command clears the log file; the component continues running.

> **Note:**
>
> You are not prompted to clear logs.

halt
Searches for the specified component and runs the **kill -9** command.

restart
Stops the component, waits for completion, then restarts the component.

start
Starts the component if it is not running.
If you specify **all** for *component*, it starts the components specified in the **pin_ctl.conf** file. For information, see "Customizing the List of Components to Start or Stop".

status
Returns the status of *component* as **Running** or **NotRunning**.

stop
Stops the component if it is running.
If you specify **all** for *component*, it stops the components specified in the **pin_ctl.conf** file. For information, see "Customizing the List of Components to Start or Stop".

Parameters for Components

You can perform an action on any of the following components:

all
Applies the action to the components specified in the **pin_ctl.conf** file. By default, the components are the following:

- Oracle Data Manager (DM)
- Email DM
- Connection Manager
- CM Master Process
- Invoice formatter

You can modify the list of components specified in the **pin_ctl.conf** file. See "Customizing the List of Components to Start or Stop".

answer
Paymentech answer simulator

batch_controller
Batch Controller

cm
Connection Manager

cm_proxy
CM Proxy

cmmp
Connection Manager Master Process

dm_eai
Enterprise Application Interface (EAI) DM

dm_email
Email DM

dm_fusa
Paymentech DM

dm_invoice
Invoice DM

dm_ldap
LDAP DM

dm_oracle
Oracle DM

dm_vertex
Vertex tax calculation DM

eai_js
EAI Java Server

formatter
Invoice formatter

pin_db_alert.pl

Use this utility to monitor the following database key performance indicators (KPIs) in Oracle databases:

- Age of event and audit tables
- Size of audit tables
- Invalid or missing procedures, triggers, or indexes

You configure this utility to alert you when one of these components has returned a certain status.

For more information, see "Monitoring Key Performance Indicators".

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/**diagnostics/pin_db_alert**

Syntax

```
pin_db_alert.pl
```

Parameters

This utility has no parameters.

pin_del_closed_accts

Use this utility to delete closed accounts from BRM. This utility calls the PCM_OP_CUST_DELETE_ACCT opcode to delete closed accounts that are older than the specified retention period. For more information, see:

- Specifying Retention Period for Closed Accounts.
- Deleting Closed Accounts

To connect to the BRM database, the **pin_del_closed_accts** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities".

Location

BRM_home/**bin**

Syntax

```
pin_del_closed_accts  -subord [-leaf]
                      -members_sharing
                      -members_billing
                      -file file_name
                      -force
                      [-verbose] [-help]
```

Parameters

-subord [-leaf]
Deletes the remaining closed nonpaying child accounts that are parents of other child accounts at the different levels of the hierarchy. It does not delete the top-level parent account in the hierarchy.
To delete the top-level parent account, run the utility without any parameters after deleting all the paying and nonpaying child accounts at different levels in the hierarchy.
Add the **-leaf** option to delete closed nonpaying child accounts at the bottom of the hierarchy.

-members_sharing
Deletes member accounts from sharing groups; for example, discount and charge sharing groups.

-members_billing
Deletes closed paying accounts in a hierarchy that are used for billing purposes.

-file *file_name*
Deletes the accounts specified in the input file, where *file_name* is the name and location of the file that contains the list of accounts for deletion. The account details in this file must be in flist format.

> **Note:**
>
> Running the **pin_del_closed_accts** utility with this parameter deletes all the accounts specified in the input file even if the accounts are not older than the retention period. When you use this parameter, ensure that the input file contains only the closed accounts that need to be deleted.

-force
Deletes closed accounts that still have active open sessions.

-verbose
Displays information about successful or failed processing as the utility runs.

-help
Displays the syntax and parameters for this utility.

Results

The **pin_del_closed_accts** utility notifies you when it successfully deletes the closed accounts and the associated customer data.

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

After you resolve the errors, you can delete the closed accounts by running the **pin_del_closed_accts** utility again.

pin_sub_balance_cleanup

Use this utility to purge expired account sub-balances from the BRM database. For more information, see "About Purging Account Sub-Balances".

> **Note:**
>
> - When you delete sub-balances from the database, events that impacted those sub-balances cannot be rerated. Ensure you no longer need the expired sub-balances before deleting them.
> - You must run this utility from the *BRM_home***/apps/pin_subscription** directory. This directory contains the **pin.conf** file that has the parameters required for this utility.

> **Note:**

Location

*BRM_home***/bin**

Syntax

```
pin_sub_balance_cleanup  -n number_of_days | -d date
                         [-t] [-verbose] [-test]
```

Parameters

-n *number_of_days*
Specifies the number of days prior to which sub-balances are deleted. For example, specify **60** to delete expired sub-balances older than 60 days.

-d *date*
The date prior to which sub-balances are deleted. Use the format *MM/DD/YYYY*. For example, specify **06/30/2003** to delete expired sub-balances older than June 30, 2003.

-t
Deletes expired sub-balances with temporary credit limits.
Use this parameter with the **-d** *date* or **-n** *number_of_days* parameter.

-test
Finds the sub-balances that meet the criteria but does not purge them from the BRM database.

-verbose
Displays information about successful or failed processing as the utility runs.

Results

The utility does not notify you if it was successful. Look in the (*BRM_home*/**apps/pin_subscription/pin_subscription.pinlog**) file to find any errors.

pin_unlock_service

Use this utility to unlock a locked CSR account. See "Unlocking a Locked CSR Account".

> **Note:**
>
> - The **pin_unlock_service** utility needs a configuration (**pin.conf**) file in the same directory from which you run the utility. The **pin.conf** file required for running this utility is located in the *BRM_home*/**apps**/**pin_unlock_service** directory.
> - Make sure that the account in the **pin.conf** file is not locked.

Location

BRM_home/**bin**

Syntax

```
pin_unlock_service
```

Parameters

This utility accepts parameters through command prompts only. After each entry, press the **Enter** key to confirm your selection. For more information, see "Unlocking a Locked CSR Account".

Results

This utility notifies you of all the results in the command console. Look in the **pin_unlock_service.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

pin_virtual_gen

Use this utility to convert event storable classes in the BRM schema to use virtual columns. After you run this utility, the **poid_type** columns of event tables in the BRM database are virtual-column enabled.

For more information, see "Generating Virtual Columns on Event Tables".

To connect to the BRM database and to specify logging information, this utility uses the **Infranet.properties** file in the directory from which you run the utility.

Specify the log level by setting the **infranet.log.level** property in the **Infranet.properties** file. The default is **1**. Valid values are **1**, **2**, and **3**. Regardless of the log level set, status messages are printed to **stdout** and to the log file. Errors are logged and printed to **stderr**.

Location

*BRM_home***/apps/pin_virtual_columns**

Syntax

```
pin_virtual_gen { { -gentasks [-execute] | -readtasks [-execute] } { [create|
pre_export|post_export|verify_types|create_types] } } | -showtasks [minID
maxID] }
```

Parameters

-gentasks create [-execute]
Generates tasks and stores them in the database.
Runs tasks after saving to the database.

-readtasks create [-execute]
Reads previously stored tasks from the database.
Runs tasks after reading from the database.

-gentasks create
Creates virtual columns and supporting columns in the BRM database.
Use this in conjunction with **showtasks** to display the tasks that will be run for creating the virtual columns before running them. The following example shows how to create the tasks for creating virtual columns, display them, and then run them:

```
pin_virtual_gen -gentasks create
pin_virtual_gen -showtasks
pin_virtual_gen -readtasks create -execute
```

-showtasks *minID maxID*
Reads corresponding tasks from the database and displays task details.
minID and *maxID* specify the tasks to show within an ID range. The command shows tasks that have an ID greater than *minID* or less than *maxID*.
All tasks are displayed when an ID range is not provided.

-gentasks pre_export [-execute]
Removes virtual columns temporarily.

-gentasks post_export [-execute]
Restores virtual columns that were temporarily removed.

-gentasks verify_types [-execute]
Verifies whether storable class type names exist in the data dictionary of the BRM database schema.

-gentasks create_types [-execute]
Creates the names of custom storable class types and stores them in the data dictionary of the BRM database schema.

-help
Displays the syntax and parameters for this utility.

Results

The utility notifies you when it runs successfully. Otherwise, look in the **vcol.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the **Infranet.properties** file.

purge_audit_tables.pl

Use this script to archive unneeded shadow objects in audit tables. Use it to:

- Generate audit table reports
- Create history tables
- Move audit tables to history tables
- Archive audit tables
- Restore archived audit tables

The different versions of shadow objects that are valid for archiving are moved to the history tables so they can be accessed for future reference. In addition, when versions of an object are removed from audit tables, all subclasses of the shadow objects are also removed automatically by the script.

> **Note:**
>
> - This script does not delete objects from the database; it only purges the object rows stored in a table.
> - To connect to the BRM database, this script needs a configuration file in the directory from which you run the script.

For more information, see "Purging Archived Audit Data" in *BRM Developer's Guide*.

Location

BRM_home/**sys/archive/oracle**

Syntax

```
purge_audit_tables.pl  report -t objects -d date -l /@connection |
                       create -t objects -l /@connection |
                       archivedirect -t objects -d date -c commit_size
-l /@connection |
                       archiveindirect -t objects -d date -c
commit_size -l /@connection |
                       renametohist -t objects -l /@connection |
                       updfromhist -t objects -d date -c commit_size -
l /@connection |
                       help
```

The following purging actions are supported:

- report Syntax
- create Syntax
- archivedirect Syntax
- archiveindirect Syntax
- renametohist Syntax
- updfromhist Syntax

report Syntax

Generates audit table reports.

This generates a file named **purge_tables.report**, which provides information about the tables for the specified objects, including the number of rows in each table that are eligible for purging, and whether history tables exist for them. You create a report to determine which mode of archiving to use for the specified object: **archivedirect** or **archiveindirect**.

```
purge_audit_tables.pl  report -t objects -d date -l /@connection
```

-t *objects*
Specifies a comma-separated list of shadow objects on which to report.
Shadow objects use an **au** prefix. For example, a change to a field marked for auditing in the **/profile** object results in the **/au_profile** shadow object.

> **Note:**
>
> Do not specify child objects for an object; they are included automatically by the script. For example, the **/au_profile/serv_extracting** object is reported on when you list the **/au_profile** object.

-d *date*
Specifies the cutoff date for purging data.
This date determines which versions of the audit object are eligible for purging. If a version of an object is valid at the cutoff date, and there is at least one older version of

the same object, the valid object is kept and all older versions are marked for purging and moved to the history tables.
The format is *YYYY*:*MM*:*DD*.

-l /@*connection*
Specifies how to connect to your database. For example:

```
perl purge_audit_tables.pl report -t au_account -d 2005:02:23 -l /@subdb
```

create Syntax

Creates empty history tables for the specified objects and their child objects.

```
purge_audit_tables.pl create -t objects -l /@connection
```

-t *objects*
Specifies a comma-separated list of objects for which to create history tables. History tables are prepended by H_ as shown in Table 17-1.

/service Object Audit Tables	/service Object History Tables
AU_SERVICE_T	H_SERVICE_T
AU_SERVICE_ALIAS_T	H_SERVICE_ALIAS_T
AU_SERVICE_EXTRACTING_T	H_SERVICE_EXTRACTING_T

> **Note:**
>
> Do not specify the child objects for a table; they are handled automatically by the script.

-l /@*connection*
Specifies how to connect to your database. For example:

```
purge_audit_tables.pl create -t au_account -l /@subdb
```

archivedirect Syntax

Archives audit tables for the specified objects and their child objects by copying the data directly to the history tables and then removing it from the audit tables.

```
purge_audit_tables.pl archivedirect -t objects -d date -c commit_size -l /
@connection
```

-t *objects*
Specifies a comma-separated list of objects to archive audit tables for.
Shadow objects use an **au** prefix. For example, a change to a field marked for auditing in the **/profile** object results in the **/au_profile** shadow object.

> **Note:**
>
> Do not specify child objects for an object; they are included automatically by the script. For example, the **/au_profile/serv_extracting** object is reported on when you list the **/au_profile** object.

-d *date*
Specifies the cutoff date for purging data.
This date determines which versions of the audit object are eligible for purging. If a version of an object is valid at the cutoff date, and there is at least one older version of the same object, the valid object is kept and all older versions are marked for purging and moved to the history tables.
The format is *YYYY*:*MM*:*DD*.

-c *commit_size*
Specifies the number of rows to save to the database at one time.

-l /@*connection*
Specifies how to connect to your database.

Sample archivedirect command

```
purge_audit_tables.pl archivedirect -t au_account -d 2005:03:29 -c
1000 -l /@subdb
```

archiveindirect Syntax

Archives audit tables for the specified objects and their child objects by copying the data first to temporary tables, then to the history tables. If successful, the old audit table data is removed.

> **Note:**
>
> Do not delete the temporary tables if the data was not copied successfully to the history tables. Errors might have occurred when the data was moved to the temporary tables from the main tables; therefore, manually transfer the data back to the main audit tables. Then, delete the temporary tables and run the script again.

```
purge_audit_tables.pl archiveindirect -t objects -d date -c
commit_size -l /@connection
```

-t *objects*
Specifies a comma-separated list of objects to archive audit tables for.
Shadow objects use an **au** prefix. For example, a change to a field marked for auditing in the **/profile** object results in the **/au_profile** shadow object.

> **Note:**
>
> Do not specify child objects for an object; they are included automatically by the script. For example, the **/au_profile/serv_extracting** object is reported on when you list the **/au_profile** object.

-d *date*
Specifies the cutoff date for purging data.
This date determines which versions of the audit object are eligible for purging. If a version of an object is valid at the cutoff date, and there is at least one older version of the same object, the valid object is kept and all older versions are marked for purging and moved to the history tables.
The format is *YYYY*:*MM*:*DD*.

-c *commit_size*
Specifies the number of rows to save to the database at one time.

-l */@connection*
Specifies how to connect to your database.

Sample archiveindirect command

```
purge_audit_tables.pl archiveindirect -t au_account -d 2005:02:23 -c 1000 -
l /@subdb
```

renametohist Syntax

Renames the specified audit tables to their corresponding history tables and re-creates the audit tables *without* any indexes. This option also creates the script files used to create, rename, rebuild, and drop indexes that were in the audit tables. You can run the following scripts manually when necessary.

- **create_index_script.sql**
- **rename_index_script.sql**
- **rebuild_index_script.sql**
- **drop_index_script.sql**

```
purge_audit_tables.pl  renametohist -t objects -l /@connection
```

-t *objects*
Specifies a comma-separated list of objects for which to rename audit tables to history tables and recreate empty audit tables.

> **Note:**
>
> You do not need to specify child objects for an object; they are included automatically by the script. For example, the **/au_profile/serv_extracting** child object is reported on if you list the **/au_profile** object.

-l *l@connection*
Specifies how to connect to your database.

Sample renametohist Command

```
purge_audit_tables.pl  renametohist -t au_account -l /@subdb
```

updfromhist Syntax

Retrieves the data for a given object and its child objects from the history tables and transfers it back to the audit tables.

```
purge_audit_tables.pl updfromhist -t objects -d date -c commit_size -
l /@connection
```

-t *objects*
Specifies a comma-separated list of shadow objects to retrieve the data from the history tables and update in the corresponding audit tables.
Shadow objects use an **au** prefix. For example, a change to a field marked for auditing in the **/profile** object results in the **/au_profile** shadow object.

> **Note:**
>
> Do not specify child objects for an object; they are included automatically by the script. For example, the **/au_profile/serv_extracting** object is reported on when you list the **/au_profile** object.

-d *date*
Specifies the cutoff date for retrieving data.
The format is *YYYY*:*MM*:*DD*.

-c *commit_size*
Specifies the number of rows to save to the database at one time.

-l *l@connection*
Specifies how to connect to your database. For example:

```
purge_audit_tables.pl  updfromhist -t au_account -d 2005:03:29 -c 1000
-l /@subdb
```

help Syntax

Displays the syntax for this script.

Results

The utility notifies you only if it encounters errors.

pin_create_server_cert

Use this utility to create an Oracle wallet and a Transport Layer Security (TLS) certificate for testing purposes. All BRM applications use this wallet to send the trusted

server certificate when a client application requests a PCM connection through TLS. For more information, see "Working with SSL/TLS Certificates and Oracle Wallets".

Location

BRM_home/**bin**

Syntax

```
pin_create_server_cert
```

Results

Creates an Oracle wallet and a server certificate in *BRM_home*/**wallet/server**.

Part II
Implementing Security

This part describes basic Oracle Communications Billing and Revenue Management (BRM) system administration tasks. It contains the following chapters:

18

Implementing System Security

Learn how to implement security in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

See also *BRM Security Guide*.

Using General System-Security Measures

You can use the usual database and operating-system security measures for the BRM system. For example, you can set up read/write/execute permissions and group permissions on files and programs.

As shipped, BRM uses encryption only for passwords. However, you can encrypt any string field. For more information, see "About Encrypting Information" in *BRM Developer's Guide*.

The following principles are fundamental to using any application securely:

- **Keep software up to date.** This includes the latest product release and any patches that apply to it.
- **Monitor system activity.** Establish who should access which system components, and how often, and monitor those components.
- **Install software securely.** For example, use firewalls, secure protocols such as SSL, and secure passwords.
- **Keep up to date on security information.** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible.

  See "Critical Patch Updates, Security Alerts and Bulletins" on the Oracle website.

Understanding the BRM Environment

When planning your BRM implementation, consider the following:

- **Which resources need to be protected?**
  - You need to protect customer data, such as credit-card numbers.
  - You need to protect internal data, such as proprietary source code.
  - You need to protect system components from being disabled by external attacks or intentional system overloads.
- **Who are you protecting data from?**

For example, you need to protect your subscribers' data from other subscribers, but someone in your organization might need to access that data to manage it. You can analyze your workflows to determine who needs access to the data; for example, a system administrator might be able to manage your system components without accessing the system data.

- **What will happen if protections on strategic resources fail?**

  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to you or your customers. Understanding the security ramifications of each resource will help you protect it properly.

Oracle Security Documentation

BRM uses other Oracle products, such as Oracle Database and Oracle WebLogic Server.

For more information, see the following documents:

- *Oracle Database Security Guide*
- *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*
- Oracle WebLogic Server documentation

Oracle documentation is available from Oracle Help Center:

http://docs.oracle.com

19

Managing Login Names and Passwords

Learn how to manage login names and passwords in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

See also "Logging Customer Service Representative Activity Events" and "Setting Up Permissions in BRM Applications".

Managing Login Names and Passwords for BRM Access

To access the BRM database, a client application must provide the following:

- An account name
- The password for that account
- The service
- The database number of the BRM database

When you install BRM, the system creates a single user account with general permission to the BRM system. The default system login name is **root.0.0.0.***n* (where *n* is your database number), and you provide its password during the BRM installation process. This system account includes two services: **admin_client** and **pcm_client**.

- BRM client applications log in to the **admin_client** service.
- Other BRM utilities and programs, such as optional service integration components, log in to the **pcm_client** service.

When you set up a production BRM system, you create additional accounts—for example, one for each of your customer service representatives (CSRs)—and associate one or more services with each account. You give each account a password and grant certain privileges to the account. For example, you might want to allow only some of your CSRs to handle payment disputes.

Before creating CSR accounts, you must use PDC to create and load a CSR package, which defines the services available to CSRs.

You also need to provide an account for any extended applications you use with BRM.

> **Note:**
>
> You cannot change the payment method of the **root** account or make it a parent or child account.

Configuring the Maximum Number of Invalid Login Attempts

You configure the maximum number of invalid login attempts by setting the **MaxLoginAttempts** business parameter.

To configure the maximum number of invalid login attempts:

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

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

   See "pin_bus_params" in *BRM Developer's Guide* for information about the utility's syntax and parameters.
3. In the file, set the value to the maximum number of login attempts:

   ```
   <MaxLoginAttempts>5</MaxLoginAttempts>
   ```

   The default value is **5**.
4. Save the file as **bus_params_act.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_act.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb".

Configuring the CM to Verify Application Logins with the Service Only

By default, the CM is configured to require a service, a login name, and a password. This provides secure access to BRM.

If only secure applications will connect to your CM, you can speed up the login process by configuring the CM to verify only the service but not require a login name or password.

To configure the CM to verify application logins with the service only:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Change the **cm_login_module** entry from **cm_login_pw001.dll** to **cm_login_null.dll**:

   ```
   - cm cm_login_module cm_login_null.dll
   ```
3. Save and close the file.

4. Stop and restart the CM.
5. Configure the applications that connect with this CM to provide only service information at log in. In the configuration file for each application, set **login_type** to **0**, and ensure a valid service is listed for **userid**.

> **Note:**
>
> CM Proxy provides another way of connecting to BRM without using a login name and password. See "Using CM Proxy to Allow Unauthenticated Log On".

Enabling Password Restriction for /service Objects

In BRM, you can use password restriction to secure the creation, modification, and deletion of **/service** objects.

Password restriction forces passwords to adhere to the following rules:

- Contain a minimum of 8 characters. It is recommended to use the longer password.
- Include at least one numeric character, one uppercase character, one lowercase character, and one special character.
- Different from the previous four passwords (NA for customer account creation and service creation).
- Should not include any part of the user ID.
- Should not contain dictionary words.
- Should not contain commonly used combinations.
- Should not contain birthday of a user or a name of the related person or other personal facts.
- Should contain minimum six digits for mobile devices.

You can configure the password restrictions using the PCM_OP_CUST_POL_VALID_PASSWD opcode.

By default, password restriction for **/service** objects is disabled in BRM. To enable it, run the **pin_bus_params** utility to change the **EnablePasswordRestriction** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable password restriction for **/service** objects:

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

   ```
   pin_bus_params -r BusParamsCustomer bus_params_customer.xml
   ```

3. In the file, set **<EnablePasswordRestriction>** to **enabled**:

   ```
   <EnablePasswordRestriction>enabled</EnablePasswordRestriction>
   ```

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

   ```
   pin_bus_params bus_params_customer.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb".

Storing Passwords in Oracle Wallet

By default, the BRM Installer stores sensitive information, such as database and account passwords, in the Oracle wallet and the BRM applications retrieves the passwords from the Oracle wallet. However, if the database and account passwords are also stored in the **Infranet.properties** and **pin.conf** configuration files, the BRM applications retrieve the passwords from the configuration files. The BRM applications automatically decrypt the encrypted passwords when retrieving them from the configuration files.

By default, the passwords in the configuration files are encrypted in the Oracle ZT PKI format. For more information, see "Encrypting Data" in *BRM Developer's Guide*.

> **Note:**
>
> To encrypt passwords for client applications or optional managers that are not part of base BRM or that are associated with customizations, use the **pin_crypt_app** utility. For details, see "About Encrypting Passwords" in *BRM Developer's Guide*.

When you encrypt a password for the Connection Manager (CM), ensure that the password adheres to the following rules:

- Contain a minimum of eight characters
- Include at least one numeric character, one uppercase character, and one special character
- Differ from the previous four passwords
- Not include any part of the user ID

Configuring Applications to Provide Login Information

BRM client applications provide login information in various ways:

- BRM Java-based applications, including Pricing Center, Customer Center, and Configuration Center, ask the user for port numbers and database names when the application starts.
- Payment Tool provides port numbers and database names in its .ini file.

To change the default login information for client applications, edit the **.ini** or configuration file or use the login dialog box.

Login Information for Java-Based Client Applications

To change most connection information for Java-based client applications, use the login dialog box, which appears when you start the application. The application uses this default information for subsequent sessions.

Login Information for Payment Tool

To change the default login information for Payment Tool:

1. Open the **C:\Users***user_name***\AppData\Local\VirtualStore\Windows\PaymentTool.ini** file.
2. Edit the login entries.
3. Save and close the file.

20

Logging Customer Service Representative Activity Events

Learn how to log CSR activity events in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Logging CSR Activity Events

You can log CSR activity events in the BRM database. The log information includes the type of activity, the date and time the CSR performed the activity, and the IP address of the client computer. This data is stored in the **/user_activity** storable class.

To log CSR activities, session-event logging must be enabled. Session-event logging is enabled by using the CM **pin.conf** file **login_audit** entry. See "Turning Off Session-Event Logging" for more information.

If you are using an external customer relationship management (CRM) application to connect to BRM, BRM also logs the external CSR user ID along with the user activity in the **/user_activity** object. You can use the external CSR ID during auditing to trace any actions performed by the external CSR in BRM. If the external CRM application does not send the external CSR user ID to BRM, BRM logs NULL values in the **/user_activity** object.

You can log CSR activities only if you have included them in the **pin_notify** file. For more information on using the **pin_notify** file for event notifications, see "Merging Event Notification Lists" in *BRM Developer's Guide*. Use the **load_pin_notify** utility to load the events specified in the **pin_notify** file into the BRM database. See "load_pin_notify" in *BRM Managing Customers*.

> **Note:**
>
> To generate a report displaying the log data, you must write your own query and GUI custom code.

You can log the following CSR activities:

- Account creation
- Account hierarchy changes
- Bill adjustments
- Bill Now operations
- Changes to billing day of month
- Changes to billing frequency

- Changes to credit card information
- Creation of charge, discount, monitor, and profile sharing groups
- Creation of deferred action schedules
- Credit limit changes
- Deletion of charge, discount, monitor, and profile sharing groups
- Disputes
- Event adjustments
- Item adjustments
- Modifications to charge, discount, monitor, and profile sharing groups
- Payments
- Password change
- Charge offer cancellation
- Charge offer modification
- Charge offer purchase
- Refunds
- Service login name change
- Settlements
- Top-ups
- Write-offs

To customize how to log CSR activity, use the PCM_OP_ACT_POL_LOG_USER_ACTIVITY policy opcode. This opcode enables you to add details for the events that need to be logged. If you add information, you need to add it to the **/user_activity** storable class by using PIN_FLD_INHERITED_INFO in the input flist.

21

Setting Up Permissions in BRM Applications

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

Topics in this document:

See also "Managing Login Names and Passwords".

Setting Up Permissions in BRM Applications

Permissions determine which tasks a user can perform with BRM applications.

A set of permissions defines a role. A role represents a set of actions that a person holding a particular job or position can perform. For more information, see "About Managing Roles".

You can restrict activities in Customer Center, Pricing Center, and other applications by assigning CSRs to a role and setting permissions for that role. For example, you can specify which CSRs can change a password, apply credits, and give refunds.

As part of setting permissions, you do the following:

- Add permissions to a role.

  > **Note:**
  >
  > For all applications except Customer Center, permission types are not granted by default. CSRs do not have access to the application or feature associated with the permission until permission is explicitly assigned. For Customer Center, some permission types are granted by default. For more information, see Customer Center Help.

- Assign an access level to each permission. For example:
  - You can specify read-only permissions to access critical data such as credit card and pricing information.
  - You can specify read-write access to customer data such as billing address and contact email ID.

- For some permissions, such as giving credits, set a minimum and maximum amount.

  When setting the minimum and maximum values for permissions that allow crediting and debiting a customer's account, ensure the value you set is appropriate for all noncurrency balance elements that apply to charge offers any customer might own. For example, if 25 is the maximum credit a CSR can issue to a customer, the CSR cannot credit more than 25 frequent flyer miles or 25 hours of service usage.

You must have proper permissions to add, change, or delete permissions. In most cases, only a person with **root** access, such as a system administrator, is granted permission to change CSR permissions.

> **Note:**
>
> If your company uses a proprietary application for administering accounts, you can write your own code to set and enforce permissions.

About Permissioning Center

Permissioning Center is a BRM application you can use to enhance security by managing the roles and permissions of BRM client tool users, such as CSRs.

You perform the following tasks using Permissioning Center:

- **Manage roles.** You can create, rename, and delete roles; add child roles; and assign CSRs to roles.
- **Manage permissions.** You can add and delete permissions.
- **Manage CSRs.** You can create CSR accounts, assign CSRs to roles, and unassign CSRs from roles.

CSRs who require access to Permissioning Center must be assigned to a role that includes permissions to use Permissioning Center. You include access permissions to a role in the same way you include any other permissions in Permissioning Center.

About Managing Roles

You use roles to configure permissions for a group of CSRs based on the tasks they must perform. For example, you can create different types of CSRs and assign them to different kinds of roles:

- **Manager CSRs** can create new roles, assign CSRs to roles, change permission settings, change credit limits, give refunds, and change account status. A manager can also validate the work that junior CSRs perform (for example, by making sure that new accounts are created correctly and have all the necessary information).
- **Junior CSRs** can check customer account balances, check and change billing information, and answer common customer questions.

You can create a role hierarchy in Permissioning Center. To do this, you create child roles and associate them with a parent role.

You organize hierarchical roles according to their permission levels. At each level above the bottom of the hierarchy, the child roles can also be parent roles. A child role inherits all permission settings that are associated with its parent role.

For example, the parent role, CSR, can also have the following child roles:

- Lead-CSR
- Junior-CSR

The child roles include all the permissions that belong to the parent role, CSR. In addition, the child roles have all the specific permissions that belong to their particular role, Lead-CSR or Junior-CSR.

About Managing Permissions in Permissioning Center

In Permissioning Center, permissions are based on roles. The role's permissions determine the specific levels of access for the role. Using Permissioning Center, you can create new CSR accounts, assign CSRs to roles, and unassign CSRs from roles. This role-based approach makes it easy to quickly grant or deny permissions to an individual CSR or a group of CSRs with a specific role.

About Permission Types for BRM Applications

You can set permissions for these applications:

- **Customer Center:** By default, every CSR has the **/customercenter/corrective_bill** permission to create corrective bills in Customer Center, and the **Actions** menu on the **Balance** tab will display the **Produce Corrective Bill** menu item. To revoke the permission, remove the **/customercenter/corrective_bill** permission for the CSR.
  You can set some permissions in both Permissioning Center and Customer Center. If you set the same permission in both, the permission you set in Customer Center takes precedence.

  For a list of Customer Center permission types, see Customer Center Help.
- **Pricing Center:** CSRs are not granted permissions to Pricing Center by default. CSRs do not have access to the application or feature associated with the permission until permission is explicitly assigned.

  For a list of Pricing Center permission types, see Table 21-1. All permission types are case sensitive.

Table 21-1 Pricing Center Permission Types

Permission Type	Provides Permission to...
/appcenter/pricingcenter	Use Pricing Center
/appcenter/provisioningtags	Using Provisioning Tags
/appcenter/ResourceEditor	Use Resource Editor
/appcenter/ZoneMapper	Use Zone Mapper
/loadpricelist/access	Load price list XML files into the BRM database

Table 21-1 (Cont.) Pricing Center Permission Types

Permission Type	Provides Permission to...
/pricingcenter/databaseaccess	Take actions that read from and write to the BRM database. This permission type controls access to the **Commit to Portal Database** and **Import - Real-time Data** commands on the **File** menu. It also controls the type of access you have to pipeline rating functionality in Pricing Center. If this permission type is set to **None**, you can only work offline in Pricing Center.
/pricingcenter/filesystemaccess	Take actions that read from and write to files. This permission type controls access to the following **File** menu commands: – **Open** – **Import - Real-time Data** – **Export - Real-time Data** – **Save** – **Save As**

- **Collections Center:** CSRs are not granted permissions to use Collections Center by default. CSRs do not have access to the application or feature associated with the permission until permission is explicitly assigned.

  For a list of Collections Center permission types, see Table 21-2. All permission types are case sensitive.

Table 21-2 Collections Center Permission Types

Permission Type	Provides Permission to...
/collectionapps/collections/updatepayment	Update promise-to-pay payment details in Collections Center.
/collectionapps/collections/newcard	Register a new credit card or direct debit account in Collections Center.
/collectionapps/collections/maskcarddetails	View all digits of a credit card or direct debit account. **Note**: If credit card tokenization is enabled, you can view only the last four digits of the credit card. See "Masking Credit Card Numbers by Using Tokens" in *BRM Configuring and Collecting Payments*.

Managing CSR Passwords

To improve security features and provide access to BRM client applications, the following password policies are included in Permissioning Center:

- **Ability to set password expiry limits.** The duration of time that a password is valid until the system prevents a user from logging in or forces the password to be changed.
- **Ability to define temporary passwords.** The ability to force CSRs to change their passwords after accessing the application the first time or after a new CSR account has been set up by an administrator.

- **Password content validation.** The ability to validate the contents of the password to ensure that certain characters are or are not included, such as numbers.

Setting CSR Account Password Status

The following are the valid password statuses for CSR accounts:

- **Temporary**
- **Normal**
- **Expires**
- **Invalid**

You can change a CSR account's password status in Permissioning Center.

To customize how passwords expire, use the PCM_OP_CUST_POL_EXPIRATION_PASSWD opcode. See "Customizing Password Expiration" in *BRM Opcode Guide*.

Unlocking a Locked CSR Account

To unlock a locked CSR account:

1. At the command prompt, run the *BRM_home***/bin/pin_unlock_service** utility.

   A menu appears.

   > **Note:**
   >
   > - The **pin_unlock_service** utility needs a configuration (**pin.conf**) file in the same directory from which you run the utility.
   > - Ensure that the account in the **pin.conf** file is not locked.

2. Press **1**.

   You are prompted to select the type of service: **admin_client** or **pcm_client**.

3. Select the service type that is associated with the account you want to unlock:
   - For **admin_client**, press **1**.
   - For **pcm_client**, press **2**.

   You are prompted to enter the login ID for the account that you want to unlock.

4. Enter the login ID and press the **Enter** key.
5. Enter a new password for the account.

   The password must satisfy the following requirements:
   - It is at least 6 characters in length.
   - It does not exceed 255 characters.
   - It contains a combination of letters and numbers.
   - It is not the same as the login ID.

6. Confirm the password to unlock the account.

   A message stating that the account has been successfully unlocked appears, followed by a menu.

7. Do one of the following:
   - To unlock another account, press **1**.
   - To exit the utility, press **2**.

Setting the Default Password Expiry Duration

To change the default password expiry duration:

1. Open the *BRM_home***/sys/cm/pin.conf** file.
2. Change the value of the **passwd_age** entry:

   ```
   - cm passwd_age 90
   ```

   The default is **90**.
3. Stop and restart the CM.

Unlocking the Locked root Account

To unlock the locked root account:

1. Go to *BRM_home***/sys/dm_oracle/data**.
2. Connect to your database using SQL*Plus:

   ```
   sqlplus pin/password@database_name
   ```

   where *database_name* is the service name or database alias of the BRM database.
3. Run the following command, which loads the stored procedure:

   ```
   @create_actlogin_unlockservice_procedures.plb
   ```
4. Run the following procedure:

   ```
   exec actlogin_unlockservice.Proc_Unlock_Service('service_type', 'user_name');
   ```

   where:
   - *service_type* is the type of service, either **/service/admin_client** or **/service/pcm_client** object. BRM stores the permission settings for each role at the **root** level in the **/service/admin_client** or **/service/pcm_client** object in the BRM database.
   - *user_name* is the root user login name.

   The **root** account is unlocked.

22

Enabling Secure Communication between BRM Components

Learn how to configure secure connections between Oracle Communications Billing and Revenue Management (BRM) applications by using protocols such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS).

> **Note:**
>
> The term *SSL* is often used to refer to either of these protocols or a combination of the two (SSL/TLS). BRM uses SSL/TLS to provide secure communication between system components.

Topics in this document:

Working with SSL/TLS Certificates and Oracle Wallets

The Oracle wallet is a password-protected container that stores SSL/TLS authentication and signature credentials, such as private keys, and certificates. You must create an Oracle wallet containing a trusted server certificate for the CM. You can create either an Oracle wallet with SSL/TLS authentication and credentials for each PCM client or one Oracle wallet whose SSL/TLS authentication and credentials are shared by a group of PCM clients. BRM C PCM and Java PCM clients use TLS 1.2 when establishing the SSL connection. BRM server components accept only TLS 1.2 connection from PCM clients.

BRM provides server and client Oracle wallets and trusted certificates that are compatible with the default cipher suite, SSL_RSA_WITH_AES_128_CBC_SHA. You can use the server and client Oracle wallets and trusted certificates and set your CM and PCM clients to the following directories:

- To use the server Oracle wallets and trusted certificate, set the CM to *BRM_home*/**wallet/server**, where *BRM_home* is the directory in which the BRM software is installed. The server Oracle wallet contains the trusted server certificate and a certificate authority (CA) certificate.
- To use the client Oracle wallet and trusted certificate, set the PCM client to *BRM_home*/**wallet/client**. The client Oracle wallet contains the trusted certificate and a certificate authority (CA) certificate.

> **Note:**
>
> By default, a sample Oracle wallet named **cwallet.sso** is installed for Java PCM clients. For more information, see "Enabling SSL/TLS for Java PCM Clients".

SSL/TLS uses signed certificates to check and verify two applications on a network. In BRM, trusted certificates are used to check and verify the identification of the CM and the PCM client. Each PCM client and CM Oracle wallet must contain a trusted certificate and a CA certificate. The CA certificate signs a certificate to make it trusted and checks certificates from other applications to make sure that they come from a trusted CA source.

A sample CA certificate is in the *BRM_home*/**wallet/root** directory. You can use the sample CA certificate, set up your own CA certificate, or use a CA certificate from a third party.

Creating an Oracle Wallet and a Server Certificate

Before you enable SSL/TLS in BRM, an Oracle wallet containing a trusted server certificate must be created for each CM, DM, and EM.

To create an Oracle wallet and server certificate:

1. Go to the *BRM_home*/**ThirdPartyApps** directory.
2. Run the **source** command on the **source.me** file:

For Bash shell:

```
source source.me.sh
```

For C shell:

```
source source.me.csh
```

3. Run the following command:

```
pin_create_server_cert
```

An Oracle wallet and trusted server certificate are created in the *BRM_home***/wallet/server** directory. The **pin_create_server_cert** script uses the sample CA certificate to sign the server certificate.

The **pin_create_server_cert** utility generates a TLS certificate for testing. Enter the certificate details as shown in Table 22-1.

Table 22-1 TLS certificate details

Certificate Details	Certificate Value
Enter Certificate DN	Enter the distinguished name that you want to use in the certificate. For example, CN=*hostname*. When the utility prompts for the distinguished name, the host name should match with the configured host name in the *dm_pointer* entry in the *BRM_home***/sys/cm/pin.conf** file. Otherwise, the TLS connection fails.
Enter RSA key size	The default value is **2048 bits**.
Enter Certificate Signature Algorithm	The default value is **sha256**.
Enter the wallet root password	Password for *BRM_home***/wallet/root**.
Enter the wallet server password	Password for *BRM_home* **/wallet/server**.

Once the wallet is created run the following command to view the generated server wallet.

```
orapki wallet display -wallet BRM_home/wallet/server
```

> **Note:**
>
> If a directory already exists in the *BRM_home***/wallet/server** directory, the utility prompts to remove or rename the directory before executing the command.

The requested certificate is successfully created.

4. (Optional) To enable two-way server and client authentication between the CM and the PCM client, create a trusted client certificate using the **orapki** utility and the sample CA certificate, and enable two-way server and client authentication for the CM.

The sample CA certificate is in the *BRM_home***/wallet/root** directory. For more information about the **orapki** utility, see *Oracle Database Advanced Security Administrator's Guide*.

For example:

a. Go to *BRM_home*/**bin**.

b. Create a certificate request by running the following command:

```
orapki wallet add -wallet BRM_home/wallet/client -keysize 1024 -
dn cn=test_client,dc=us,dc=oracle,dc=com -pwd password
```

c. Export the certificate request to a file by running the following command:

```
orapki wallet export -wallet BRM_home/wallet/client -dn
cn=test_client,dc=us,dc=oracle,dc=com -request BRM_home/wallet/
client/ccreq.txt -pwd password
```

where the **-request** parameter specifies the file name.

d. Create a trusted certificate by running the following command:

```
orapki cert create -wallet BRM_home/wallet/root -request
BRM_home/wallet/client/ccreq.txt -cert BRM_home/wallet/client/
ccert.txt -validity 3650 -pwd password
```

where the **-validity** parameter specifies the number of days that the certificate is valid.

e. Add the client certificate into the client wallet by running the following command:

```
orapki wallet add -wallet BRM_home/wallet/client -user_cert -
cert BRM_home/wallet/client/ccert.txt -pwd password
```

f. Enable two-way server and client authentication for the CM. For more information on how to enable two-way authentication between server and client, see "Enabling SSL/TLS in Connection Managers".

Using the orapki Utility to Create Oracle Wallets and Certificates

You can use the **orapki** utility to create Oracle wallets and certificates and to import certificates into Oracle wallets. The **orapki** utility is in the *BRM_home*/**bin** directory.

For more information about the **orapki** utility, see *Oracle Database Advanced Security Administrator's Guide*.

To create an Oracle wallet by using the **orapki** utility:

1. Go to the *BRM_home*/**bin** directory.
2. Run the following command:

```
orapki wallet create -wallet wallet_location -auto_login -pwd password
```

where:

- *wallet_location* is the directory in which the Oracle wallet is to be created.

> **Note:**
>
> Ensure that there are no blank spaces within the *wallet_location* directory name or path.

- *password* is the password used to make changes to the Oracle wallet.

  Passwords for the **orapki** utility must adhere to the following rules:

  – Contain a minimum of eight characters

  – Include at least one numeric character, one uppercase character, and one special character

  – Differ from the previous four passwords

  – Not include any part of the user ID

> **Note:**
>
> For security reasons, Oracle recommends that you do not specify the password at the command line. Instead, enter the password when prompted to do so.

All SSL/TLS Oracle wallets must have the **auto_login** parameter enabled. The **auto_login** parameter opens the wallet without the need for a password. If you wish to make changes to the wallet, a password must be provided.

BRM-Supported Cipher Suites

Cipher suites contain authentication, encryption, and MAC algorithms that are used to protect information during key exchange, bulk encryption, and message authentication. BRM lists the cipher suites it supports, in order of preference.

During the SSL/TLS handshake between the PCM client and the CM, the PCM client sends the supported cipher suite list to the CM. The CM selects a cipher suite and returns its selection to the PCM client.

Table 22-2 lists the SSL/TLS cipher suites that are supported for C and C++ based BRM programs such as the CM and Oracle DM.

Table 22-2 Supported TLS Cipher Suites for BRM C/C++ Programs

TLS Program	IANA Cipher Suite Name	BRM C/C++ Cipher Name
TLSv1.2	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLSv1.2	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLSv1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLSv1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Table 22-3 lists the SSL/TLS cipher suites that are supported for Java-based BRM programs such as EAI Manager.

Table 22-3 Supported TLS Cipher Suites for BRM Java Programs

TLS Program	IANA Cipher Suite Name	BRM Java Cipher Name
TLSv1.2	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLSv1.2	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLSv1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLSv1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLSv1.3	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLSv1.3	TLS_AES_256_GCM_SHA384	TLS_AES_128_GCM_SHA256

Enabling or Disabling SSL/TLS for BRM Components

By default, SSL/TLS is enabled for all BRM components.

To enable or disable SSL/TLS for all BRM components at one time:

1. Go to the *BRM_home***/setup/scripts** directory.
2. Do one of the following:
   - To enable SSL/TLS between the clients and the CM, and disable SSL/TLS between the CM and DM/EM, run the following command:

     ```
     perl sslConfig.pl 2
     ```
   - To enable SSL/TLS, run the following command:

     ```
     perl sslConfig.pl 1
     ```
   - To disable SSL/TLS, run the following command:

     ```
     perl sslConfig.pl 0
     ```

Enabling or Disabling SSL/TLS for BRM Clients

By default, SSL/TLS is enabled for all BRM clients.

To enable or disable SSL/TLS for a BRM client:

1. Open the *BRM_client_home/application_Name***/Infranet.properties** file in a text editor.

   where:

   - *BRM_client_home* is the directory in which the BRM client application is installed.
   - *application_Name* is the name of the BRM client application.
2. Do one of the following:

- To enable SSL/TLS, set the following entry to **true**:

  ```
  infranet.pcp.ssl.enabled=true
  ```

- To enable SSL/TLS, set the following entry to **false**:

  ```
  infranet.pcp.ssl.enabled=false
  ```

Enabling SSL/TLS in Connection Managers

By default, SSL/TLS is enabled in the CMs. If you disabled SSL/TLS during the BRM server installation, you can enable SSL/TLS in the CMs.

> **Note:**
>
> If you have multiple CMs on the same machine or if you are creating a new CM, each CM needs its own Oracle wallet. See "Creating an Oracle Wallet and a Server Certificate".

To enable SSL/TLS in a CM:

1. Open the *BRM_home*/**sys/cm/pin.conf** file in a text editor.
2. Add the following entry:

   ```
   - cm enable_ssl 1
   ```

3. (Optional) To enable two-way server and client authentication between the CM and the PCM client, add the following entry:

   ```
   - cm ssl_auth 2-way
   ```

   For information on how to create a trusted server certificate to use when enabling two-way authentication, see "Creating an Oracle Wallet and a Server Certificate".

4. (Optional) If your server Oracle wallet is not in the default directory (*BRM_home*/**wallet/server**), add the following entry:

   ```
   - cm wallet wallet_location
   ```

   where *wallet_location* is the full path to the directory in which your server Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

5. (Optional) To add cipher suites, do one of the following:

   - To add one cipher suite, add the following entry:

     ```
     - cm  cipher cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

For example:

```
- cm cipher SSL_RSA_WITH_AES_256_SHA384
```

- To add multiple cipher suites, separate each cipher suite by a comma.

  For example:

```
- cm cipher
SSL_RSA_WITH_AES_256_GCM_SHA384,SSL_RSA_WITH_AES_128_GCM_SHA256
```

6. Save and close the file.
7. Stop and restart the CM.

Enabling SSL/TLS in Data Managers

By default, SSL/TLS is enabled in the DMs. If you disabled SSL/TLS during the BRM server installation, you can enable SSL/TLS in the DMs.

> **Note:**
>
> If you have multiple DMs on the same machine or if you are creating a new DM, each DM needs its own Oracle wallet. See "Creating an Oracle Wallet and a Server Certificate".

To enable SSL/TLS in a DM:

1. Open the *BRM_home***/sys/***data_manager***/pin.conf** file in a text editor, where *data_manager* is the folder for the DM you want to enable secure communication for.
2. Add the following entry:

   ```
   - dm enable_ssl 1
   ```

3. (Optional) To enable two-way authentication between the CM and the DM, add the following entry:

   ```
   - dm ssl_auth 2-way
   ```

   For information on how to create a trusted server certificate to use when enabling two-way authentication, see "Creating an Oracle Wallet and a Server Certificate".

4. (Optional) If your server Oracle wallet is not in the default directory (*BRM_home***/wallet/server**), add the following entry:

   ```
   - dm wallet wallet_location
   ```

   where *wallet_location* is the full path to the directory in which your server Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

5. (Optional) To add cipher suites, do one of the following:
   - To add one cipher suite, add the following entry:

     ```
     - dm  cipher cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

     For example:

     ```
     - cm cipher SSL_RSA_WITH_AES_256_SHA384
     ```
   - To add multiple cipher suites, separate each cipher suite by a comma.

     For example:

     ```
     - dm cipher SSL_RSA_WITH_AES_256_GCM_SHA384,SSL_RSA_WITH_AES_128_GCM_SHA256
     ```
6. Save and close the file.
7. Stop and restart the DM.

Enabling SSL/TLS for C and C++ PCM Clients

By default, the ability to use SSL/TLS with C and C++ PCM clients is enabled. If you disabled SSL/TLS during the BRM clients installation, you can enable SSL/TLS for the C and C++ PCM clients.

To enable SSL/TLS for C and C++ PCM clients:

1. Open the **pin.conf** file of the PCM client application in a text editor.
2. Add the following entry:

   ```
   - nap enable_ssl 1
   ```
3. (Optional) To enable two-way server and client authentication between the CM and C and C++ PCM clients, add the following entry:

   ```
   - nap ssl_auth 2-way
   ```
4. (Optional) If your client Oracle wallet is not in the default directory, add the following entry:

   ```
   - nap wallet wallet_location
   ```

   where *wallet_location* is the full path to the directory in which your client Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.
5. (Optional) To add cipher suites for C and C++ PCM clients, do one of the following:
   - To add one cipher suite, add the following entry:

     ```
     - nap cipher cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

For example:

```
- nap cipher SSL_RSA_WITH_AES_256_SHA384
```

- To add multiple cipher suites, separate each cipher suite by a comma.

  For example:

```
- nap cipher
SSL_RSA_WITH_AES_256_GCM_SHA384,SSL_RSA_WITH_AES_128_GCM_SHA256
```

6. Save and close the file.
7. Stop and restart the CM.

Enabling SSL/TLS for Java PCM Clients

By default, the ability to use SSL/TLS with Java PCM clients is enabled. When you install a Java client application, a sample Oracle wallet named **cwallet.sso** is installed in the directory in which you install the Java client application. If you disabled SSL/TLS during the BRM clients installation, you can enable SSL/TLS for the Java PCM clients.

> **Note:**
>
> Only administrators with write permissions can make changes to the **Infranet.properties** file.

To enable SSL/TLS for Java PCM clients, do the following for each Java PCM client:

1. Open the Java **Infranet.properties** file in a text editor.

   Table 22-4 lists the default location of the **Infranet.properties** file for each Java PCM client.

Table 22-4 BRM Client Infranet.properties File Default Locations

Client Name	Directory Path
Business Configuration Center	Windows 8.1 and Windows 10 — C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Business Operations Center	Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\BusinessOperationCenter\RevenueAssuranceCenter\lib\Infranet.properties
Collections Configuration	C:\Program Files (x86)\Portal Software\CollectionsConfiguration\Lib\Infranet.properties
Customer Center	Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\CustomerCenter\lib\Infranet.properties
Developer Center	C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
IP Address Administration Center	C:\Program Files (x86)\Portal Software\IPAddressAdministrationCenter\Infranet.properties
Number Administration Center	C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Permissioning Center	C:\Program Files (x86)\Portal Software\PermissioningCenter\lib\Infranet.properties

Table 22-4 (Cont.) BRM Client Infranet.properties File Default Locations

Client Name	Directory Path
Pricing Center	Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\PricingCenter\lib\Infranet.properties
SIM Administration Center	C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Suspense Management Center	C:\Program Files (x86)\Portal Software\SuspenseManagementCenter\lib\Infranet.properties
Voucher Administration Center	C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties

2. Search for the following line:

   ```
   - infranet.pcp.ssl.enabled=false
   ```

3. Change **false** to **true**.

   ```
   - infranet.pcp.ssl.enabled=true
   ```

4. (Optional) To enable two-way server and client authentication between the CM and Java PCM clients, add the following entry:

   ```
   - infranet.pcp.ssl.client_auth = true
   ```

5. (Optional) If the Java PCM client Oracle wallet is not in the default location, add the following entry:

   ```
   - infranet.pcp.ssl.wallet.location=wallet_location/wallet
   ```

   where *wallet_location* is the full path to the directory in which your Java PCM client Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

6. (Optional) If your Java PCM client Oracle wallet name is different from the sample Java PCM client Oracle wallet name, add the following entry:

   ```
   - infranet.pcp.ssl.wallet.filename=wallet_name.sso
   ```

   where *wallet_name* is the name of your Java PCM client Oracle wallet.

7. (Optional) Change the timeout setting for the SSL/TLS handshake between the CM and Java PCM clients by setting the **infranet.pcp.ssl.handshake.timeout** entry to the appropriate number of milliseconds:

   ```
   - infranet.pcp.ssl.handshake.timeout=timeout_in_milliseconds
   ```

   The default timeout is 30000 milliseconds.

8. (Optional) To add cipher suites for Java PCM clients, do one of the following:

   - To add one cipher suite, specify the name of the cipher suite in the following entry:

     ```
     - infranet.pcp.ssl.handshake.ciphersuites=cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

For example:

```
- infranet.pcp.ssl.handshake.ciphersuites=TLS_RSA_WITH_AES_256_GCM_SHA384
```

- To add multiple cipher suites, separate each cipher suite by a comma.

  For example:

```
-
infranet.pcp.ssl.handshake.ciphersuites=TLS_RSA_WITH_AES_128_CBC_SHA,TLS_
RSA_WITH_AES_256_GCM_SHA384
```

9. Save and close the file.
10. Verify that the *BRM_home*/**jars/oraclepki.jar**, *BRM_home*/**jars/osdt_cert.jar**, *BRM_home*/**jars/osdt_core.jar**, and *BRM_home*/**jars/commons-logging-1.2.jar** files are in the CLASSPATH.

Enabling SSL for Web Start Deployment

You must enable SSL for the following Web Start deployments:

- Customer Center
- Pricing Center
- Suspense Management Center

To enable SSL for a Web Start deployment:

1. Package your SSL wallet files and **Infranet.properties** file into a separate JAR file (for example, named **wallet.jar**) and digitally sign it.

   The following shows a sample directory structure for the **wallet.jar** file:

```
wallet.jar
    |_ com/oracle/wallet/cwallet.sso
    |_ com/oracle/wallet/ewallet.p12
     _ Infranet.propeties
```

2. Copy the JAR file to the *Client_Application_home*/**3plibs** directory, where *Client_Application_home* is the directory in which you installed Customer Center, Pricing Center, or Suspense Management Center on your Web server. For example: C:/Program Files/Apache Group/Tomcat/webapps/ROOT/CustomerCenter.
3. Create a new Java Network Launch Protocol (JNLP) file, such as **brmwallet.jnlp**, and copy it to the *Client_Application_home* directory in which the *client_application*_**en.jnlp** file is located.

   where client_application is one of the following:

   - **CustomerCenter**
   - **PricingCenter**
   - **SuspenseManagement**

   The following shows sample content in a JNLP file:

```
<?xml version="1.0" encoding="utf-8"?>
    <jnlp spec="1.0+" codebase="__CODE_BASE_URL__" href="brmwallet.jnlp">
        <information>
            <title>SSL Wallet Certificates</title>
```

```
            <vendor>Oracle Corporation</vendor>
        </information>
        <security>
            <all-permissions/>
        </security>
        <resources>
            <jar href="3plibs/wallet.jar" />
        </resources>
        <component-desc/>
    </jnlp>
```

4. In the *Client_Application_home*/*client_application*_**en.jnlp** file, add these two JNLP file extensions under the **resources** element:

```
<resources>
   <extension name="brmwallet" href="brmwallet.jnlp"/>
</resources>
```

5. Update the location and name of the wallet file by adding the following entries to the **Infranet.properties** file:

```
- infranet.pcp.ssl.wallet.location=wallet_location/wallet
- infranet.pcp.ssl.wallet.filename=wallet_name.sso
```

   where:

   - *wallet_location* is the wallet path within the JAR file that you created, such as **com/oracle/wallet**.

     > **Note:**
     >
     > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

   - *wallet_name* is the name of the wallet file, such as **cwallet.sso**.

6. Launch the Web Start for Customer Center, Pricing Center, or Suspense Management Center and connect to a SSL-enabled BRM system.

Enabling SSL/TLS for Java Server Processes

By default, the ability to use SSL/TLS with Java server processes is enabled in BRM. If you disabled SSL/TLS during the BRM installation, you can enable SSL/TLS for Java server processes.

> **Note:**
>
> Only administrators with write permissions can make changes to the **Infranet.properties** file.

To enable SSL/TLS for Java server processes:

1. Open the **Infranet.properties** file for your Java server process in a text editor.

Table 22-5 lists the default location of the **Infranet.properties** file for each Java server process.

Table 22-5 Java Server Process Infranet.properties File Default Locations

Process Name	Directory Path
EAI Java Server or eai_js	*BRM_home*/sys/eai_js/Infranet.properties
BRM invoice formatter	*BRM_home*/sys/formatter/Infranet.properties
pin_job_executor utility	*BRM_home*/apps/pin_job_executor/Infranet.properties
Kafka DM	*BRM_home*/sys/dm_kafka/Infranet.properties

2. Do one of the following:
   - To enable TLSv1.2, set the following property to **true**:

     ```
     - infranet.pcp.ssl.enabled=true
     ```

   - To enable TLSv1.3, set the following property to **true**:

     ```
     - infranet.pcp.ssl.enable_tlsv13=true
     ```

3. (Optional) To enable two-way server and client authentication between the CM and the Java server process, add the following entry:

   ```
   - infranet.pcp.ssl.client_auth = true
   ```

4. (Optional) If the Java server process Oracle wallet is not in the default location, add the following entry:

   ```
   - infranet.pcp.ssl.wallet.location=wallet_location/wallet
   ```

   where *wallet_location* is the directory in which your Java server process Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

5. (Optional) If your Java server process Oracle wallet name is different from the sample the Java server process Oracle wallet name, add the following entry:

   ```
   - infranet.pcp.ssl.wallet.filename=wallet_name.sso
   ```

   where *wallet_name* is the name of your Java server process Oracle wallet.

6. (Optional) To add cipher suites for Java server process, do one of the following:
   - To add one cipher suite, specify the name of the cipher suite in the following entry:

     ```
     - infranet.pcp.ssl.handshake.ciphersuites=cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

     For example:

     ```
     - infranet.pcp.ssl.handshake.ciphersuites=TLS_RSA_WITH_AES_256_GCM_SHA384
     ```

- To add multiple cipher suites, separate each cipher suite by a comma.

  For example:

  ```
  -
  infranet.pcp.ssl.handshake.ciphersuites=TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA
  _WITH_AES_256_GCM_SHA384
  ```

7. Save and close the file.

Enabling SSL/TLS in Connection Manager Master Processes

By default, SSL/TLS is enabled in CMMPs. If you disabled SSL/TLS during the BRM server installation, you can enable SSL/TLS in CMMPs.

If you have multiple CMMPs on the same machine or if you are creating a new CMMP, each CMMP needs its own Oracle wallet. See "Creating an Oracle Wallet and a Server Certificate".

When enabling SSL/TLS in CMMP, all the CMs listed in the **redirects** parameter in the CMMP's **pin.conf** file must be SSL/TLS enabled.

To enable SSL/TLS in a CMMP:

1. Open the *BRM_home*/**sys/cmmp/pin.conf** file in a text editor.
2. Add the following entry:

   ```
   - cm enable_ssl 1
   ```

3. (Optional) To enable two-way server and client authentication between the CMMP and the PCM client, add the following entry:

   ```
   - cm ssl_auth 2-way
   ```

   For information on how to create a trusted server certificate to use when enabling two-way authentication, see "Creating an Oracle Wallet and a Server Certificate".

4. (Optional) If your server Oracle wallet is not in the default directory (*BRM_home*/**wallet/server**), add the following entry:

   ```
   - cm wallet wallet_location
   ```

   where *wallet_location* is the full path to the directory in which your server Oracle wallet resides.

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

5. (Optional) To add cipher suites, do one of the following:

   - To add one cipher suite, add the following entry:

     ```
     - cm  cipher cipher_suite
     ```

     where *cipher_suite* is the name of the cipher suite.

     For example:

     ```
     - cm cipher SSL_RSA_WITH_AES_256_SHA384
     ```

- To add multiple cipher suites, separate each cipher suite by a comma.

  For example:

  ```
  - cm cipher SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_RSA_WITH_AES_256_GCM_SHA384
  ```

6. Save and close the file.
7. Stop and restart the CMMP.

Enabling SSL/TLS for Payment Tool

By default, the ability to use SSL/TLS with Payment Tool is disabled. When you install Payment Tool, a sample Oracle wallet named **cwallet.sso** is installed in the **C:\Program Files\Common Files\Portal Software\wallet\client** directory.

To enable SSL/TLS for Payment Tool on Windows:

> **Note:**
>
> - Only administrators with write permissions can make changes to the **PaymentTool.ini** file.
> - After enabling SSL/TLS in the **PaymentTool.ini** file, run Payment Tool in the administrator mode.

To enable SSL/TLS for Payment Tool:

1. Create a directory for the Payment Tool Oracle wallet (*wallet_location*).

   > **Note:**
   >
   > Ensure that there are no blank spaces within the *wallet_location* directory name or path.

2. Copy the **cwallet.sso** file from the **C:\Program Files\Common Files\Portal Software\wallet\client** directory to *wallet_location*.
3. Open the **C:\Windows\PaymentTool.ini** file in a text editor.
4. Add the following entry:

   EnableSSL=1
   SSLWallet=*wallet_location*

5. (Optional) To add cipher suites for Payment Tool, do one of the following:
   - To add one cipher suite, add the following entry:

     SSLCipher=*cipher_suite*

     where *cipher_suite* is the name of the cipher suite. For example:

     ```
     SSLCipher=SSL_RSA_WITH_RC4_128_MD5
     ```

   - To add multiple cipher suites, separate each cipher suite by a comma. For example:

```
SSLCipher=SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_RSA_WITH_RC4_128_MD5
```

By default, BRM uses SSL_RSA_WITH_AES_128_CBC_SHA. For information on cipher suites supported by BRM, see "BRM-Supported Cipher Suites".

6. Save and close the file.

Enabling SSL/TLS with Custom Applications

For custom applications, use the login flist as input to the PCM_CONTEXT_OPEN opcode.

Table 22-6 lists the fields and values that you must set in the login flist to enable SSL/TLS.

Table 22-6 Login flist Fields for SSL/TLS

Field	Data Type	Value
PIN_FLD_ENABLE_SSL	PIN_FLDT_INT	To enable SSL/TLS, use **1**. To disable SSL/TLS, use **0**.
PIN_FLD_SSL_CIPHER	PIN_FLDT_STR	Cipher suite, or cipher suites separated by a comma.
PIN_FLD_SSL_WALLET	PIN_FLDT_STR	Path to the Oracle wallet directory.

Enabling SSL/TLS for Paymentech DM

By default, the TLS connection between Paymentech (dm_fusa) and answer simulator is disabled. However, by default, the TLS connection between CM and Paymentech is enabled.

To enable SSL/TLS for dm_fusa:

1. Open the *BRM_home***/sys/dm_fusa/pin.conf** file in a text editor.
2. Add the following entry:

```
- dm_fusa fusa_tls_enabled 1
```

3. Save and close the file.
4. Stop and restart the Paymentech DM.

Enabling SSL/TLS for Paymentech Answer Simulator

By default, SSL/TLS is disabled for answer simulator.

To enable SSL/TLS for answer simulator in **pin.conf**:

1. Open the *BRM_HOME*/**apps/fusa_server/pin.conf** file in a text editor.
2. Add the following entry:

```
- answer answer_tls_enabled 1
```

3. Save and close the file.
4. Stop and restart the Paymentech answer simulator.

Verifying Server Host Name

When SSL/TLS is enabled for secure communication between the BRM components, depending on whether the component is acting as the SSL/TLS client or as the SSL/TLS server, BRM verifies the host name configured in the SSL/TLS certificate in the following manner:

- When the C and C++ PCM clients connect to the CM as an SSL/TLS server, the server host name configured in the server certificate from the CM will be verified with the **cm_ptr** parameter in the client's **pin.conf** file.
- When the Java PCM clients connect to the CM as an SSL/TLS server, the server host name configured in the server certificate from the CM will be verified with the host name in the connection URL.
- When the CM is the client to a DM as the SSL/TLS server, the server host name configured in the server certificate from the DM will be verified with the **dm_pointer** parameter in the CM's **pin.conf** file.
- When the CM is the client to an EM as the SSL/TLS server, the server host name configured in the server certificate from the EM will be verified with the **em_pointer** parameter in the CM's **pin.conf** file.
- When CMMP is used for connection between the Java PCM clients and the CM, the CMMP redirects the connection to the CM. The server host name configured in the server certificate from the CM will be verified with the host name in the connection URL.

If the host name does not match, the connection is terminated.

SSL/TLS Client Certificate Authentication

BRM server components such as the DM and EM maintain a list of trusted certificates and trusted CA certificates in the server wallet. When SSL/TLS two-way authentication is enabled for the server component, you can verify the client certificate containing the host name details with the list of trusted certificates and trusted CA certificates present in the server wallet. After SSL/TLS handshake, if the certificate is used by a trusted CA, the connection is allowed. If the certificate is not used by a trusted CA, the connection is terminated.

If you want to allow only the certificates used by Oracle CA and not any other CA, ensure that other CAs are not present in the server wallet.

Creating Debugging Logs for SSL/TLS

You can use the PIN_SSL_DEBUG environment variable to log messages that contain debugging information related to SSL/TLS and to generate SSL/TLS trace logs.

To create debugging logs for SSL/TLS:

1. Set the PIN_SSL_DEBUG environment variable with the hexadecimal value described in Table 22-7. For example, to log messages that contain debugging information for SSL/TLS initialization, set PIN_SSL_DEBUG to **0x0001**:

```
PIN_SSL_DEBUG     0x0001
```

2. To create multiple logs and trace files, use the `OR` function on the hexadecimal values described in Table 22-7. For example, to log messages that contain debugging information related to SSL/TLS initialization and to generate SSL/TLS trace files, set PIN_SSL_DEBUG to **0x0031**:

```
PIN_SSL_DEBUG      0x0031
```

Table 22-7 PIN_SSL_DEBUG Values

Value	Description
0x0001	Logs messages that contain debugging information related to SSL/TLS initialization.
0x0002	Logs messages that contain debugging information related to read.
0x0004	Logs messages that contain debugging information related to write.
0x0030	Generates SSL/TLS trace files for the CM and C and C++ PCM clients. **Note:** Trace logs are generated in the working directories of the respective binaries.

23
Managing Closed Accounts

Learn how to manage closed accounts in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Specifying Retention Period for Closed Accounts

You can specify the number of months to retain closed accounts in BRM by setting the **ClosedAcctsRetentionMonths** parameter in the **customer** instance of the **/config/business_params** object.

To specify the retention period for closed accounts:

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

   ```
   pin_bus_params -r BusParamsCustomer bus_params_customer.xml
   ```

3. Set the **ClosedAcctsRetentionMonths** entry to the number of months that you want to retain the closed accounts:

   ```
   <ClosedAcctsRetentionMonths>number_of_months</ClosedAcctsRetentionMonths>
   ```

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

   ```
   pin_bus_params bus_params_customer.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. See "pin_multidb" in *BRM System Administrator's Guide*.

Deleting Closed Accounts

After a closed account passes the specified retention period, you can delete it from BRM by using the **pin_del_closed_accts** utility.

To delete closed accounts:

1. Go to the *BRM_home*/**apps/pin_billd** directory.
2. Do the following as appropriate:

> **Note:**
>
> To delete all closed child accounts in a hierarchy and sharing groups, run all of these commands in this order.

- To delete all closed nonpaying child accounts at different levels in a hierarchy:

  ```
  pin_del_closed_accts -subord -leaf
  pin_del_closed_accts -subord
  ```

- To delete member accounts from sharing groups:

  ```
  pin_del_closed_accts -members_sharing
  ```

- To delete paying child accounts at different levels in a hierarchy:

  ```
  pin_del_closed_accts -members_billing
  ```

  > **Note:**
  >
  > Run this command for each paying account in a hierarchy. For example, if there are two paying accounts in a hierarchy, run this command twice to delete both paying accounts.

3. To delete all remaining closed accounts, including the top-level parent account in the hierarchy:

   ```
   pin_del_closed_accts
   ```

4. To delete specific closed accounts by using a file:

   > **Note:**
   >
   > Run the **pin_del_closed_accts -file** command only if you want to delete specific accounts, but ensure that you use this command with care.

   pin_del_closed_accts -file *file_name*

   For example:

   ```
   pin_del_closed_accts  -file  closed_accts_list.txt
   ```

   The utility deletes the accounts specified in the input file. You must provide the account details in flist format. For example:

   ```
   0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 1

   1     PIN_FLD_POID           POID [0] 0.0.0.1 /account 123 0

   0 PIN_FLD_RESULTS       ARRAY [1] allocated 20, used 1

   1     PIN_FLD_POID           POID [0] 0.0.0.1 /account 234 0
   ```

For more information about the **pin_del_closed_accts** utility and parameters, see "pin_del_closed_accts".

Part III
Improving Performance

This part describes how to improve performance for an Oracle Communications Business and Revenue Management (BRM) system. It contains the following chapters:

24

Improving BRM Performance

Learn how to improve performance in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

- BRM Account and Rating Performance Considerations
- About Benchmarking
- BRM Performance Diagnosis Checklist
- Troubleshooting Poor Performance

See also "Improving Connection Manager Performance" and "Improving Data Manager and Queue Manager Performance".

BRM Account and Rating Performance Considerations

Certain aspects of basic BRM functionality can affect performance; other aspects have no effect:

- The number of accounts does not affect performance.
- There is no performance difference when using different payment methods, such as invoice or credit card.
- BRM client applications, such as Billing Care, have little impact on system performance.
- Cycle events are rated faster than usage events.
- Performance decreases when accounts own a large number of charge offers.
- Performance may decrease when the database contains a large number of ratable usage metrics (RUMs). In addition to computing the RUM for the specific event, BRM computes the RUMs for all base types of that event. For example, if you configured a RUM for the **/a/b/c/d** event, BRM also computes RUMs configured for the **/a/b/c** and **/a/b** events. You can increase performance by removing unused RUMs from your database.

About Benchmarking

To determine the best possible performance for your system, you need to identify the desired transaction capacity at each tier and ensure that the system handles several times that capacity. The maximum capacity threshold or the transaction capacity threshold can be determined by running benchmark scenarios.

The primary goal is to achieve full utilization of the database system. This is best accomplished by measuring system performance as the following operations are carried out:

- Increase the load to get maximum throughput.
- Increase the number of DM back ends to get maximum RDBMS utilization for a given workload.

- Increase database utilization for a given number of DM back ends.
- Slowly reduce the load to keep the same performance but with faster response time.
- Multiple iterations of the above steps.

The general process for benchmarking is:

1. Create a workload on the system. The best choice to do this is by running your own program. The second best choice is to use the ITM-C workloads.
2. Measure the results. Most programs need a ramp-up period of 200 seconds to reach a steady-state condition during which actual measurements should take place. Running the program for 10 to 20 minutes should produce the same results as running the program for hours, with the exception of the amount of disk space used. If you run the system for a long time, indexes might become imbalanced and need to be rebuilt, especially before billing.

   Use monitoring tools for the systems on which BRM is running to determine system load and identify performance issues. Be sure to turn monitoring tools off for your final test runs. When you start the system, turn on level 3 debugging in all BRM processes and ensure that there are no error messages while running the benchmark programs. When there are no more errors, turn off logging.
3. Monitor the hardware, operating system, and BRM utilization.

BRM Performance Diagnosis Checklist

When troubleshooting performance, use this checklist to help you look for problems. For more information, see the following:

- Monitoring Your BRM System
- Resolving Problems in Your BRM System

You can also use this checklist to gather information that Support needs when diagnosing trouble tickets. If you submit a performance issue to technical support, you should also include the following:

- All applicable error log files (for example, log files—or portions of log files—for the CM, DM, and client applications).
- Operating system settings such as maximum shared memory segment size and number of processes. Provide a full list:
  - Solaris: **/etc/system**
  - Linux: **/etc/sysctl.conf**
  - `lsattr -E -l sys0`
- Administrative tools for managing systems:
  - Solaris: **Admintool**
  - Linux: **Webmin** and **Easilix**
- The **pin.conf** files for CMs, DMs, and clients.
- For Oracle, the **init.ora** file.

Describe the Problem

- What part of the system is experiencing the problem?
- What operation or application is running (for example, billing or credit card processing)?
- What is the actual and expected performance?
- What appears to be the problem?
- What are the error messages?

Describe the Configuration

Provide Support information about the following configurations:

- Hardware Configuration
- Operating System Configuration
- BRM Configuration
- Network Configuration
- Database Server Configuration
- Oracle Configuration

Hardware Configuration

For each system in the configuration:

- What is the manufacturer, model, number, and types of CPUs, and amount of RAM?
- What is the swap size?

For the database server system:

- What is the RAID level?
- How many disks, and what is their size?
- How are logical volumes configured?

Operating System Configuration

- What is the operating system version?
- What are the operating system settings for maximum shared memory segment size, number of processes, and so forth?
- Which patches have been applied?

BRM Configuration

- Which release of BRM are you using?
- Which systems do the following components run on? Which systems have multiple components, and which components run on multiple systems? How are the **pin.conf** files configured?
  - CMMP

  - CM Proxy
  - CM
  - DM
  - BRM client applications
  - Custom applications
  - Billing utilities
- Which BRM operations are slow?
- Which PCM_OPs are those slow operations associated with? (This can be found by using log level 3.)
- What is the estimated number of accounts in the database?
- What is the average number of charge offers per account?
- What is the largest quantity of charge offers owned by one account?
- What percentage of accounts use which payment method (for example, credit card or invoice)?
- What is the estimated number of events in the database?

Network Configuration

- How are the systems connected (for example, 10BaseT or 100BaseT)?
- Are separate networks used for each DM database connection?

Database Server Configuration

- What are the index and data file sizes?
- What are the database hot spots?
- What is the disk layout?
- What is the assignment of tablespaces to logical devices?
- Are disk volumes used?
- Are redo logs on their own disk?

Oracle Configuration

- What is the Oracle version?
- How is the **init.ora** file configured?

  The following **init.ora** Oracle parameters are particularly important.

  - **db_block_buffers**
  - **shared_pool_size**
  - **use_aysnc_io**
  - **db_block_size**
  - **max_rollback_segments**
  - **processes**

  - **dml_locks**
  - **log_buffer**

  Compare how your parameters are configured to those in the example BRM performance configurations.

  - Does the SGA roughly equal half the physical RAM?
  - What are the sizes and number of rollbacks?
  - Is check-pointing or archiving enabled?
  - Index and table fragmentation?
  - Number of extents, next extent size?
  - Run the query **select index_name from user_indexes** to view indexes. Check the indexes vs. columns in the WHERE clause.
  - Which optimizer option is being used (CHOOSE or RULE)?

Describe the Activity

- Are there any messages in any error logs (CM, DM, application)?
- Are there any operating system or database system error messages?
- Are there any bad blocks?
- Are you using any nonstandard resources, custom code (especially in the CM), or debugging aids such as writing log records to files that might result in contention or bottlenecks?
- Is there enough free swap space?
- What is the CPU utilization on servers used for BRM processes?
- Database system:
  - What are I/Os per disk per second, size of disk queues, disk service time, and percent of time waiting for I/O?
  - What is the CPU utilization on the database system?

Troubleshooting Poor Performance

When troubleshooting poor performance, first consider the following:

- Under-configured hardware.
- Inefficient table layout.
- Database bottlenecks.
- Inefficient custom application code.
- Repeated runtime errors resulting from configuration problems.

In addition, you can look for different problems depending on whether CPU utilization is high or low.

Low Performance with High CPU Utilization

If performance is low and CPU utilization is high, or if there are performance spikes, there is probably a configuration or indexing issue. Check the following:

- Hardware limitations.
- Table/volume layout.
- Spin count is too high.
- Lack of proper indexes. This can show up as very high CPU utilization with no other apparent problems except for a high number of processes. Find which columns are being accessed in the operation being performed and ensure that they are properly indexed.
- Not enough database buffers.
- Swapping.
- Kernel parameters too low.

Low Performance with Low CPU Utilization

If performance is low and CPU utilization is low, check for a bottleneck between different system tiers (for example, between the DM and the database).

- Use the database monitoring tools to analyze the performance of the database system.
- Use SQL tracing and timing to check for inefficient application code.
- Check for an under-configured BRM system, which could be one of the following:
  - CM Proxy with a low number of children.
  - DMs with a low number of back ends.
  - System logging level is too high.

Monitor the DM system utilization and Oracle system utilization and tune the number of DM back ends accordingly. A good starting point for DM back-end numbers is eight times the number of processors.

For more information, see "Improving Data Manager and Queue Manager Performance".

Quick Troubleshooting Steps

- Run quick timing tests by using the **testnap** utility with **op_timing** turned on to ping each CM and DM (with the PCM_OP_TEST_LOOPBACK opcode). If the operations are relatively slow, it indicates a problem in the basic configuration.
- Run the system with a log level of DEBUG on the CM and DM and analyze log files.
- Check for network collisions and usage data.
- Check if you have logging (debugging) turned on in the CM. Logging is good for troubleshooting, but it should not be turned on in a production environment because it reduces performance.

- Performance parameters in **pin.conf** files should be large enough to handle the load. The most likely problems are in the DM entries.
- Check if you have enough DM back ends to handle your transaction load.
- Try putting tables and indexes on different disks.
- Check the size of redo and rollback logs and database configuration parameters.
- Send a few **kill -USR1** commands to the DMs and CMs that seem to be having problems. This causes them to dump their state to the BRM error log files. Snapshots should be up to 20 minutes apart. These log files may contain information that indicates the nature of the problem.
- Turn on SQL tracing and analyze query plans. Look for full table scans. Ensure that indexes are on the appropriate columns for the query being run. Especially verify for any customizations.
- Turn on the **timed_statistics** parameter. Look for unusually long execution times for SQL commands.
- Monitor hardware activity:
  - On Linux, and Solaris systems, use **vmstat**, **netstat**, and **sar**.
  - Drill down to the storage device level by using **sar** with the **-d** parameter. This should help you find the source of the problem.

    > **Note:**
    >
    > If the file systems are configured from logical volumes that are comprised of physical disks, different file systems could be sharing the same underlying disk. It is important to unravel who owns what in order to isolate potential contention (waiting on I/O).

- Problems such as intermittent daemon failures can be indicated by core files. Try the following command to locate them:

  ```
  % find BRM_home -name core -exec file {} \;
  ```

  If there are no core files, try turning on maximal debugging. You do not want to do this for very long, especially on a production system, because the log files fill up rapidly.

  ```
  % pin_ctl stop cm
  % setenv CMAP_DEBUG to 0x1331f3
  % setenv CM_DEBUG to 0x0001
  % setenv cm_loglevel to 3
  % pin_ctl start cm
  ```

  System level tracing can also be useful:

  ```
  # ps -ef | grep cm
  # truss -p cm_pid
  ```

25

Improving Connection Manager Performance

Learn how to improve Connection Manager (CM) performance in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Increasing CM Login Performance

To increase CM login performance, see the following topics:

Using CM Proxy to Allow Unauthenticated Log On

Use CM Proxy to provide unauthenticated connections to BRM, typically for providing connections for incoming mail messages. Connections made through CM Proxy do not require you to log on, which increases performance.

> **Note:**
>
> Only connections that do not require authentication should use CM Proxy.

To increase security, you can restrict the types of operations performed by CM Proxy by specifying the opcodes that perform allowed operations.

To use CM Proxy:

1. Open the CM Proxy configuration file (*BRM_home*/**sys/cm_proxy/pin.conf**).
2. Configure CM Proxy according to the guidelines in that file, and save the file.

   The following are some of the more important entries:

   - Use the **oplist** entry to specify the opcodes that can be performed by CM Proxy.
   - Use the **allowed** entry to specify the hosts that can use CM Proxy.
   - Use the **queue manager** entries to manage front-end and back-end connections. See "Improving Data Manager and Queue Manager Performance".

- Use the **standard connection** entries to connect to a CM or CMMP (Connection Manager Master Process). See "About Connecting BRM Components".

3. Start CM Proxy. See "Starting and Stopping the BRM System".
4. Open the configuration file for each application you want to use CM Proxy. See "About Configuration Files".
5. Change the **cm_ptr** entry to point to the machine running CM Proxy and change the **login_type** entry to **0** (no login name or password required).

Turning Off Session-Event Logging

By default, the CM writes a session-event object for each client connection, but you can suppress the creation of these session objects if you do not need these records of logins. You can turn off session-event logging for some CMs for better performance while keeping this feature for other CMs dedicated to applications that require session objects, such as the mail and terminal servers.

To turn off session-event logging:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value for the **login_audit** entry to **0** and ensure that the entry is not commented.
3. Stop and restart the CM.

Turning Off the Checking of Logons and Passwords

When an application tries to log on to BRM, the CM verifies the service specified by the application, asks for a login name, and verifies the login password. To improve performance, set up the CM to not ask for a login name or password. See "Configuring the CM to Verify Application Logins with the Service Only".

To turn off login name and password verification:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value for the **cm_login_module** entry to read:

   ```
   - cm   cm_login_module   ./cm_login_null.extension
   ```

   where *extension* is the file type specific to your operating system: **so** for Solaris, Linux. For example:

   ```
   - cm   cm_login_module   ./cm_login_null.so
   ```

3. Stop and restart the CM.

Load Balancing CMs

You can use two methods for balancing the load among multiple CMs:

- You can use a CMMP to provide additional reliability and to balance the load among multiple CMs. See "Using Connection Manager Master Process to Improve Performance".

- You can provide a simple failover system for connections to the CM by giving the application a list of CMs on the system. If the first CM in the list is unavailable, the application tries the next CM.

  For example:

  ```
  - nap   cm_ptr   ip   cm_host1   11960
  - nap   cm_ptr   ip   cm_host1   11961
  ```

  If the CMs are on separate machines:

  ```
  - nap   cm_ptr   ip   cm_host1   11960
  - nap   cm_ptr   ip   cm_host2   11960
  ```

  > **Note:**
  >
  > You can point to multiple CMMPs rather than to individual CMs. If the CM provided by the first CMMP is not available, the application asks for a CM from the second CMMP.

  For example:

  ```
  - nap   cm_ptr   ip   CMMP_host1   11959
  - nap   cm_ptr   ip   CMMP_host2   11959
  ```

Specifying the Number of Connections to CMs

The **cm_max_connects** entry in the CM configuration file (**pin.conf**) tells the CM how many client applications can connect at one time. If client applications are having trouble connecting to the database, you can increase the number of connections. Performance degrades when too many CMs are running on the same machine, depending on the system load.

The maximum number of connections is 1000.

> **Note:**
>
> Normally, **cm_max_connects** is commented out. This type of connection is best handled by CM Proxy. See "Using CM Proxy to Allow Unauthenticated Log On".

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

   ```
   - cm   cm_max_connects   300
   ```

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

Setting the CM Time Interval between Opcode Requests

When a client application requests a connection, the CM spawns a child process to handle the connection. The child process or thread communicates with the client application by receiving requests in the form of opcodes.

By default, the child process or thread waits infinitely for each opcode request, but you can set a time interval after which the child process or thread terminates if no request has arrived.

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit the **cm_timeout** entry.

   ```
   - cm cm_timeout 5
   ```

   The value of this entry specifies the time interval in minutes.
3. Save and close the file.
4. Stop and restart the CM.

26

Using Connection Manager Master Process to Improve Performance

Learn how to improve performance in your Oracle Communications Billing and Revenue Management (BRM) system by using the Connection Manager Master Process (CMMP).

Topics in this document:

About CMMP

The Connection Manager Master Process (CMMP) routes connections from a single BRM client to multiple CMs. See "About Connection Manager Master Processes (CMMPs)" in *BRM Concepts*.

To specify which CMs to route connections to, you list the CMs in the CMMP configuration file. (See "Setting Up a CMMP".) You can also specify how the CMMP chooses a CM:

- By default, the CMMP chooses a CM randomly from the list of CMs.
- You can specify that the CMMP choose CMs sequentially from the first entry to the last, and then back to the first (known as *round-robin* selection).

The CMMP does not validate whether the CMs it points to are running. If the CMMP sends a transaction to a nonfunctioning CM, the client can tell that the CM is offline and try to connect to a different CM or CMMP. Therefore, you should include multiple **cm_ptr** entries to the same CMMP or to more than one CMMP.

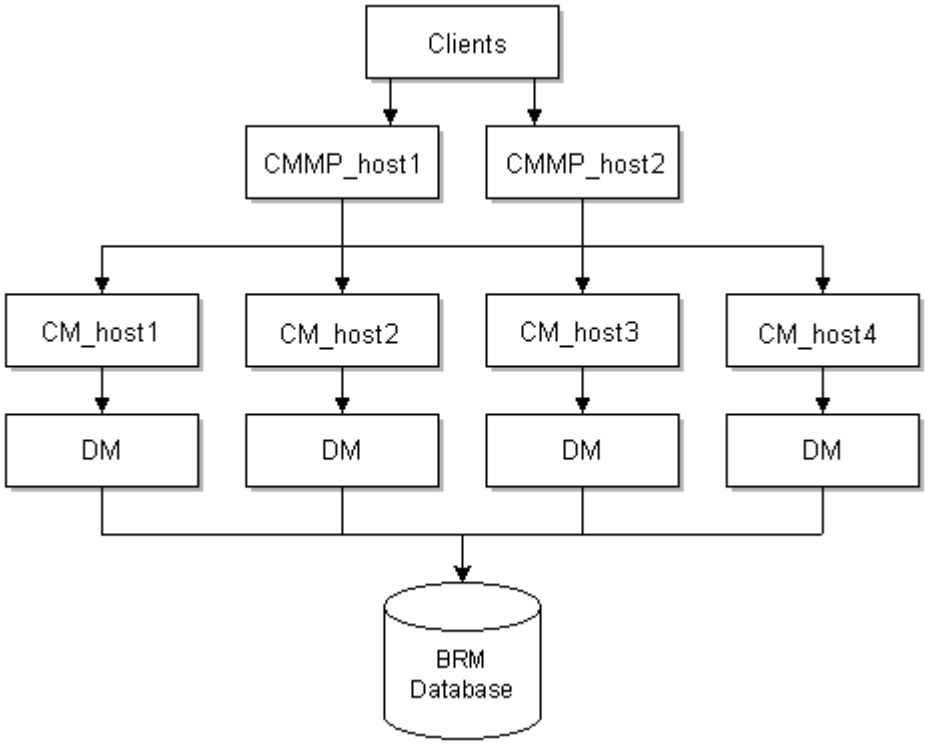
Because it is the client's responsibility to ensure a connection to a CM, there must be at least one entry in the CMMP **pin.conf** file for each CM to which it is connected.

Sample CMMP Configuration

In this example shown in Figure 26-1:

- There are two CMMP host machines, CMMP_host1 and CMMP_host2.
- There are four CM host machines, CM_host1, CM_host2, CM_host3, and CM_host4.
- Each CM points to its own DM.

Figure 26-1 Sample CMMP Configuration



Client configuration file

Using the connection parameters shown below, the clients attempt to connect to all CMs through both CMMPs:

```
- nap cm_ptr ip CMMP_host1 11959
- nap cm_ptr ip CMMP_host1 11959
- nap cm_ptr ip CMMP_host1 11959
- nap cm_ptr ip CMMP_host1 11959
- nap cm_ptr ip CMMP_host2 11959
- nap cm_ptr ip CMMP_host2 11959
- nap cm_ptr ip CMMP_host2 11959
- nap cm_ptr ip CMMP_host2 11959
```

CMMP configuration files

The following redirect entries are required for client-to-CM connectivity. The parameters should be the same for both CMMP configuration files:

```
- cm redirect - CM_host1 11960
- cm redirect - CM_host2 11960
- cm redirect - CM_host3 11960
- cm redirect - CM_host4 11960
```

If the client and the host are on different domains, the redirect entry must include the full host and domain name or IP address for the CMMP Port.

```
- cm redirect - CM_host1.example.com 11960
- cm redirect - CM_host2.example.com 11960
- cm redirect - CM_host3.example.com 11960
- cm redirect - CM_host4.example.com 11960
```

```
- cm redirect -  198.51.100.1 11960
- cm redirect -  198.51.100.2 11960
- cm redirect -  198.51.100.3 11960
- cm redirect -  198.51.100.4 11960
```

If the CMs are on the same host, the port numbers must be unique:

```
- cm redirect - CM_host1 11961
- cm redirect - CM_host1 11962
- cm redirect - CM_host1 11963
- cm redirect - CM_host1 11964
```

> **Note:**
>
> Ensure you also set the **- cm cmmp_algorithm** parameter.

Setting Up a CMMP

To set up a CMMP:

1. Open the CMMP configuration file (*BRM_home***/sys/cmmp/pin.conf**).
2. Configure the CMMP according to the guidelines in the file:
   - Use the **redirect** entry to list the CMs on your system. For example:

     ```
     - cm redirect - CM_host1 11960
     - cm redirect - CM_host2 11960
     - cm redirect - CM_host3 11960
     - cm redirect - CM_host4 11960
     ```

   - Use the **cmmp_algorithm** entry to specify whether the CMMP chooses CMs randomly (the default) or sequentially.
3. Save and close the file.
4. Start the CMMP. See "Starting and Stopping the BRM System".
5. Open the configuration file for each application you want to use CMMP.
6. Change the **cm_ptr** entry in the client application **pin.conf** file to point to the machine running the CMMP.
7. Save and close each configuration file you change.

27

Improving Data Manager and Queue Manager Performance

Learn how to improve Data Manager (DM) performance in your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

About Queuing-Based Processes

Queuing improves system performance by lowering the number of connections to the database. This reduces the number of processes and therefore reduces the system load required to handle the connections.

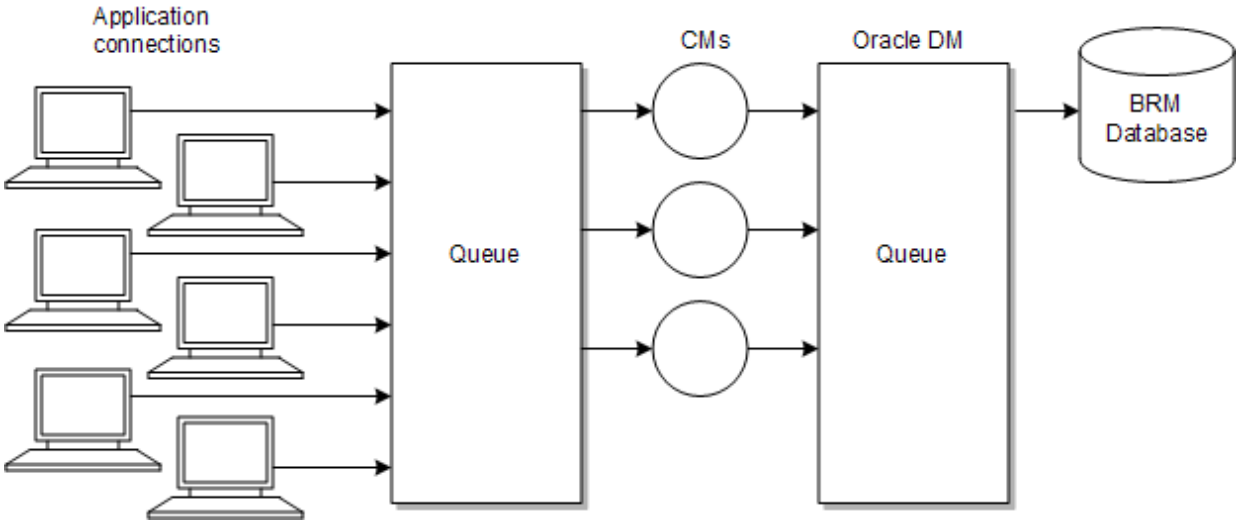
Queuing is used in two different types of system components:

- The CM Proxy and Web Interface daemons use queuing to connect incoming client connections to CMs. In this case, queuing reduces the number of client connections to CMs.
- All DMs use queuing internally. Front-end processes pass requests and data through a queue to back-end processes. In this case, queuing reduces the number of connections to the database.

CMs and Connection Manager Master Processes (CMMPs) do not use queuing.

Figure 27-1 shows an example of where queuing takes place in the BRM system architecture. Note that queuing occurs in two locations.

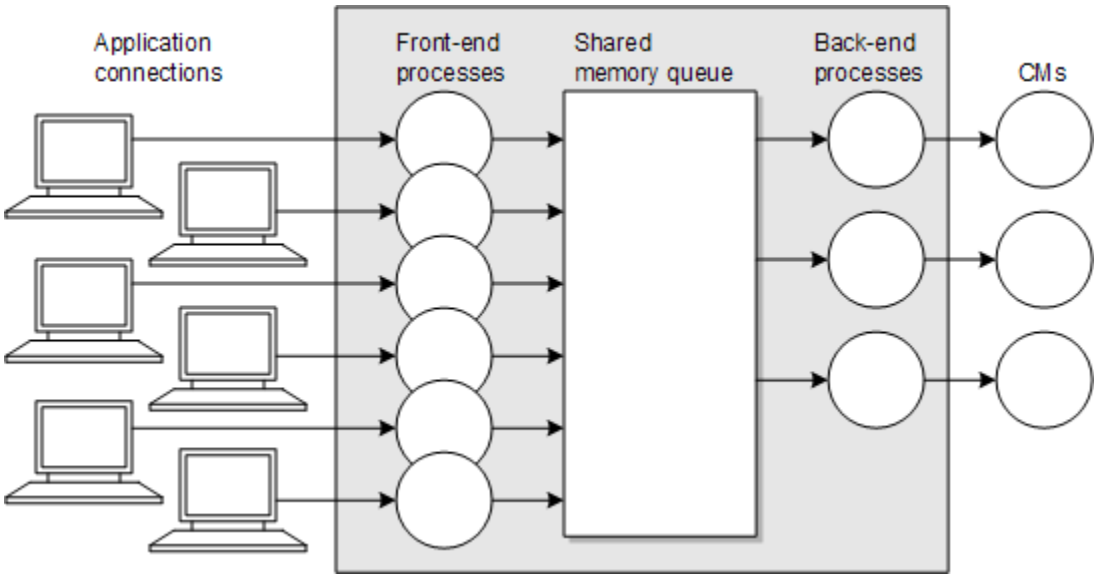
Figure 27-1 BRM Queuing Locations



Example of Queuing in a Client-to-CM Connection

Figure 27-2 shows a daemon running on a system. Front-end processes pass the connections to a shared-memory queue where the connections wait for available back ends. The back ends connect to CMs.

Figure 27-2 CM Client Connection Queuing



Configuring DM Front Ends and Back Ends

You configure DM performance by specifying the number of front-end and back-end processes and the amount of shared memory the DM uses.

> **Note:**
>
> Queue Manager (QM) components, such as LDAP Manager, use the same types of configuration entries, but they have different names. For example, instead of **dm_max_fe**, the entry is named **qm_max_fe**. The functionality is the same.

Use the following DM and QM **pin.conf** entries to tune performance:

- **dm_n_fe:** Specifies the number of DM front-end processes.
- **dm_n_be**: Specifies the maximum number of DM back-end processes.
- **dm_max_per_fe**: Specifies the maximum number of connections for each front end.
- **dm_trans_be_max**: Specifies the maximum number of back ends that can be used for processing.
- **dm_init_be_timeout**: Specifies the time, in seconds, that the DM waits for the DM back-end startup process to complete.
- **dm_trans_timeout**: Specifies the time in minutes that DM back-end processes wait for the next opcode in a transaction.

To change one or more of these parameters:

1. Open the DM configuration file (*BRM_home*/**sys/dm_oracle/pin.conf**).
2. Change the configuration entry associated with the parameter. For tuning guidelines, see the topics following this procedure. For the syntax of each configuration entry, follow the guidelines in the configuration file.
3. Save and close the file.
4. Stop and restart the DM.

> **Note:**
>
> Besides configuring the number of connections for best performance, remember to keep the number of connections within the terms of your database license agreement.

Ratio of Front Ends to Back Ends

Oracle recommends that the total number of front ends (specified in the **dm_n_fe** and **dm_max_per_fe** entries) should be two to four times the number back ends (specified in the **dm_n_be** entry).

In this example, the total number of front ends is 64 (4 times 16), which is 4 times the number of back ends.

```
- dm dm_n_fe 4
- dm dm_max_per_fe 16
- dm dm_n_be 16
```

Providing Enough Front-End Connections

If connection errors occur between the CM and DM, increase the values in the **dm_n_fe** and **dm_max_per_fe** entries. If there are not enough front ends, BRM reports an error. For example:

```
DMfe #3: dropped connect from 194.176.218.1:45826, too full
W Thu Aug 06 13:58:05 2001 dmhost dm:17446 dm_front.c(1.47):1498
```

Check the **dm**_*database*.**log** and **dm**_*database*.**pinlog** files for errors.

You must have enough DM front-end connections (number of processes times the number of connections for each process) to handle the expected number of connections from all of the CMs. Otherwise, you will see errors when the applications cannot connect to BRM.

Connections might be required for the following:

- One connection for each CM Proxy thread.
- One connection for each Web interface thread, plus one additional connection if customers create accounts with a Web interface.
- One connection for each billing application thread plus one additional connection for the main search thread.
- Two connections for each instance of Billing Care.

The maximum number of connections each front-end process can handle depends on the activity of the connection and, on multi-processor machines, the processor speed. For intensive connections, such as a heavily utilized terminal server, a front end might be able to handle only 16 connections. For intermittent connections, such as through certain client tools, a single front end can handle 256 or 512 connections. For systems that use a combination of these activities (for example, real-time processing with some client tool activity), you can configure an intermediate value for the maximum connections per front end.

For a given number of connections, if you have too many front ends (too few connections for each front end), the DM process uses too much memory and there is too much context switching. Conversely, if you have too few front ends (too many connections for each front end), the system performs poorly.

Determining the Required Number of Back Ends

You configure the number of back ends to get maximum performance from your system, depending on the workload and the type of BRM activity. Here are some guidelines for various activities:

- **Authentication/authorization**: For processing terminal server requests, which consist of many single operations without an explicit transaction, size the number of back ends to handle the traffic and leave the percentage of back ends available for transactions at the default value (50%).

  For example:

  ```
  -dm dm_n_be 48
  -dm dm_trans_be_max 24
  ```

  Normally, however, you configure the DM to perform a variety of tasks.

- **Account creation**: This activity uses one transaction connection for a long time and a second regular connection intermittently. You must provide two back ends for each of the accounts you expect to be created simultaneously. Your system might lock up if you do not have enough back ends. You can leave the percentage of back ends available for transactions at the default.

  For example:

  ```
  -dm dm_n_be 48
  -dm dm_trans_be_max 46
  ```

  The example above allows you to have 23 account creation sessions active simultaneously.

- **Billing**: Because all billing operations are transactions, ensure there is at least one back end capable of handling transactions for each billing program thread, plus one additional back end for the main thread searches.

  For example:

  ```
  -dm dm_n_be 24
  -dm dm_trans_be_max 22
  ```

  The example above allows you to have approximately 20 billing sessions (children) active simultaneously.

In general, if you need rapid response times, reduce the number of transactions waiting to be processed by adding more back ends, devoting a larger number of them to transactions, or both. For example, try increasing the number of back ends to 3 to 4 times the number of application processes. For performance, dedicate at least 80% of the back ends to processing transactions. For heavy updating and inserting environments, especially when billing is running, dedicate all but two of the back ends to transaction processing.

For example:

```
-dm dm_n_fe 4
-dm dm_max_per_fe 16
-dm dm_n_be 24
-dm dm_trans_be_max 22
```

If you configure too many back ends, the DM process uses too much memory and there is too much context switching. Conversely, if you have too few back ends, the system performs poorly and the network is overloaded as terminal servers retry the connection.

> **Note:**
>
> If there are not enough DM back ends, BRM may stop responding without reporting an error message.

On small BRM systems, where you might use a single DM for multiple activities, you can calculate the peak requirements for a combination of those activities and size the back ends accordingly. For example, you might need 32 connections for authentication and authorization and another 8 for the Web interface. If you run billing at hours when the rest of the system is relatively quiet, you do not need additional back ends.

> **Note:**
>
> The number of back ends is independent of the number of front ends. That is, front ends are not tied to particular back ends because requests are transferred via the shared memory queue.

To help gauge the correct number of back ends, monitor database utilization. If it is under-utilized, you can increase the number of back ends.

Determining the Maximum Number of Back Ends Dedicated to Transactions

The maximum number of back ends dedicated to transactions (specified in the **dm_trans_be_max** entry) should be at least 80% of the number of back ends specified in the **dm_n_be** entry. For heavy transaction loads, such as when running billing, use a value that is 2 less than the **dm_n_be** entry. For example:

```
- dm dm_n_be    48
- dm dm_trans_be_max    46
```

> **Note:**
>
> You cannot specify more transaction back ends than there are total back ends.

Setting the DM Time Interval between Opcode Requests

By default, the DM back-end processes wait an infinite amount of time for each opcode request, but you can set a time interval after which the DM back-end terminates if no opcode request has arrived. The following DM **pin.conf** entry specifies the maximum amount of time to wait, in minutes, for an opcode call before cancelling the transaction:

```
- dm dm_trans_timeout 4
```

> **Note:**
>
> To have DM back-end processes wait forever, set this entry to **0**.

Setting How Long the DM Waits for the Background Startup Process to Complete

The DM back-end startup process connects to the BRM database and initializes the BRM data dictionary into DM memory. By default, the DM waits 60 seconds for the DM

back-end startup process to complete before timing out. You can modify how long the DM waits by adding the **dm_init_be_timeout** entry to the DM **pin.conf** file.

```
- dm dm_init_be_timeout 60
```

> **Note:**
>
> This entry is not included in the default DM **pin.conf** file, so you must add it manually.

Setting DM Shared Memory Size

BRM queuing increases system performance by lowering the number of connections to the database. This reduces the number of processes, which reduces the system load required to handle the connections. All DMs use shared memory for internal queuing. Front-end processes pass connections through a shared-memory queue to back-end processes.

To specify DM shared memory, you use the following entries in the DM configuration file (**pin.conf**):

- **dm_shmsize**: Specifies the size of the shared memory segment, in bytes, that is shared between the front ends and back ends. The maximum allowed value of **dm_shmsize** in the DM's **pin.conf** file is **274877905920** bytes (256 GB).
- **dm_bigsize**: Specifies the size of shared memory for “big" shared memory structures, such as those used for large searches (with more than 128 results) or for PIN_FLDT_BUF fields larger than 4 KB.

  The maximum allowed value of **dm_bigsize** in the Data Manager's (DM's) **pin.conf** file is now **206158429184** bytes (192 GB). The value of **dm_bigsize** should always be set less than the value of **dm_shmsize**.

To specify DM shared memory:

1. Open the DM configuration file (*BRM_home***/sys/dm_oracle/pin.conf**).
2. Change the configuration entry associated with each parameter. For tuning guidelines, see the discussions following this procedure. For the syntax of each configuration entry, follow the guidelines in the configuration file.

   > **Note:**
   >
   > You may have to increase the **shmmax** kernel parameter for your system. It should be at least as large as the **dm_shmsize** entry in the DM configuration file on any computer running a DM. Otherwise, the DM will not be able to attach to all of the shared memory it might require and BRM will fail to process some transactions. See your vendor-specific system administration guide for information about how to tune the **shmmax** parameter.

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

> **Note:**
>
> Besides configuring the number of connections for best performance, remember to keep the number of connections within the terms of your database license agreement.

Determining DM Shared Memory Requirements

The amount of shared memory required by a DM depends on:

- **Number of front ends**: Each front end takes about 32 bytes of shared memory for its status block.
- **Number of connections per front end**: Each connection to a front end takes at least one 8-KB block of shared memory.
- **Number of back ends**: Each back end takes about 32 bytes of shared memory for its status block.
- **Size and type of DM operations**: Most of the shared memory used is taken by DM operations, and particularly by large searches. For example:
  - Running the **pin_ledger_report** utility.
  - Running searches that return large numbers or results.
  - Using large value maps. Allocate 1 MB of memory in the **dm_bigsize** entry for every 3000 lines in a value map.

  Operations that read objects, read fields, or write fields and involve a large BUF field can also be significant, but they are rare. Normal operations take 0 to 16 KB above the 8-KB-per-connection overhead.

You can also reduce the requirements for shared memory by using the PCM_OP_STEP_SEARCH opcode instead of the PCM_OP_SEARCH opcode.

You should monitor the shared memory usage and the transaction queues for each DM. See "Monitoring DM Shared Memory Usage" and "Monitoring DM Transaction Queues".

How BRM Allocates Shared Memory for Searches

The **dm_shmsize** entry sets the total size of the shared memory pool. The **dm_bigsize** entry sets the size of the portion of the shared memory reserved for “big" shared memory structures. Therefore, the memory available to front ends, back ends, and normal (not “big") operations is the value of the **dm_shmsize** entry minus the value of the **dm_bigsize** entry.

For example, with these entries, the shared memory available to normal operations is 25165824:

```
- dm dm_shmsize  33554432
- dm dm_bigsize  8388608
```

> **Note:**
>
> The value for **dm_shmsize** must be a multiple of 1024. The value of **dm_bigsize** must be a multiple of 8192.

To allocate memory for a search, BRM uses regular shared memory until the search returns more than 128 results. At that point, BRM reallocates the search to use the memory set aside for “big" structures. When allocating this type of memory, BRM doubles the size of the initial memory requirement in anticipation of increased memory need.

For example, consider a search that returns the POIDs of accounts that need billing. For 100,000 accounts, the memory allocated to the search is as follows:

- Memory used by “big" structures: 3.2 MB.

  The 3.2 MB figure is derived by taking the size of a POID and the anticipated number of accounts read in a billing application and then doubling the amount of memory as a safety margin.

  100,000 x 16 x 2 = 3,200,000 (3.2 MB), which is rounded up to a multiple of 8192. For example, **dm_bigsize** would be set to 3203072 or 391 x 8192.

  As a general rule, **dm_shmsize** should be approximately 4 to 6 times larger than **dm_bigsize**.

- Memory used by “small" structures: 4 MB.

  This memory is allocated for the following:

  – 2 MB for the result account POIDs (100,000 accounts x 20-byte chunks).

  – 2 MB for the POID types (100,000 accounts x 20-byte chunks).

- Total memory use: 7.2 MB.

Shared Memory Guidelines

DM shared memory is limited to 512 MB. Billing applications, internet telephony, and searching can affect DM shared memory requirements. It is usually best to start with a lower amount of shared memory to keep system resource usage minimal.

Shared memory for database servers can be from 512 MB for medium scale installations to several GB or more for the largest installations, depending upon activities. Some experimentation is necessary because more than 1 GB may not provide a performance increase, especially if there is a lot of update activity in the BRM database.

This example shows Solaris 2.6 kernel tuning parameters (for **/etc/system**) for the database server:

```
set bufhwm=2000
set autoup=600
set shmsys:shminfo_shmmax=0xffffffff
set shmsys:shminfo_shmseg=32
set semsys:seminfo_semmns=600
set semsys:seminfo_semmnu=600
set semsys:seminfo_semume=600
set semsys:seminfo_semmsl=100
forceload:drv/vxio
forceload:drv/vxspec
```

> **Note:**
>
> This example of a Solaris kernel configuration essentially sets the maximum shared memory limit to infinity. When this setting is used, the system can allocate as much RAM as required for shared memory.

Reducing Resources Used for Search Queries

You can increase performance for search queries that retrieve objects with multiple rows from the database (for example, account searches for multiple customers) by setting the value of the **dm_in_batch_size** entry in the DM configuration file (**pin.conf**).

BRM interprets the value of **dm_in_batch_size** as the number of matching rows to retrieve in one search. When you start a search, BRM runs n+1 searches, where n is the number of searches performed to retrieve the number of rows set in **dm_in_batch_size**. For example, if **dm_in_batch_size** is set to 25 and the search retrieved 100 matching rows, five searches were performed ([25 x 4]+1). The default setting is 80, indicating that BRM runs two searches to retrieve up to 80 matching rows. The maximum value is 160.

To preserve resources, you set the value in **dm_in_batch_size** to correlate to the size of the data set being searched. To increase performance when searching large data sets, you increase the number of retrieved rows in **dm_in_batch_size**. The larger the value set in **dm_in_batch_size**, the more resources are used to perform the search query. For example, if a typical user search query returns 10 rows from the database and **dm_in_batch_size** is set to 100, more resources than necessary are being used to complete the search.

Load Balancing DMs

The **dm_pointer** entry in the CM configuration file (**pin.conf**) tells the CM which DM to connect to. Having pointers to several DMs provides reliability because the system will switch to another DM if one DM fails.

You can ensure a more even load among the available DMs by adding several identical pointers to each DM, even if the DMs are on the same machine. When a CM receives a connection request, it chooses one of the pointers at random. Or, you can increase the load on a particular DM by increasing the relative number of pointers to that DM.

For example, if you have two DMs and you want to ensure that most activity goes to one with the most powerful hardware, make three or four pointers to that DM and only one or two to the other DM. When new child CM processes or threads are created, more of them are configured to point to the first DM:

```
- cm  dm_pointer  0.0.0.1  ip  127.0.0.1 15950
- cm  dm_pointer  0.0.0.1  ip  127.0.0.1 15950
- cm  dm_pointer  0.0.0.1  ip  127.0.0.1 15950
- cm  dm_pointer  0.0.0.1  ip  127.0.0.3 11950
```

Optimizing Memory Allocation during Database Searches

You can configure the Oracle DM to optimize memory allocation during database searches by using the **extra_search** entry in the DM configuration file. When this entry is set, the Oracle DM performs an extra search in the BRM database to calculate the number of database objects meeting the search criteria and then allocates the optimal amount of memory for the results.

> **Note:**
>
> Performing the extra search slows database search performance.

To optimize memory allocation by performing an extra search:

1. Open the Oracle DM configuration file (*BRM_home*/**sys/dm_oracle/pin.conf**).
2. Change the **extra_search** entry to **1**:

   ```
   - dm extra_search 1
   ```

3. Save and close the file.
4. Stop and restart the Oracle DM.

Improving BRM Performance during Database Searches

Oracle databases can access tables that have nonbitmap indexes by performing an internal conversion from ROWIDs to bitmap and then from bitmap back to ROWIDs. This internal conversion process can significantly decrease BRM performance when a large number of rows are queried.

To increase search performance, Oracle recommends that you prevent the database from using bitmap access paths for nonbitmap indexes. To do so, add the following parameter to your database's **init.ora** file or **spfile**, and then restart your database:

```
_b_tree_bitmap_plans=false
```

Increasing DM CPU Usage

If the CPU usage on a DM machine reaches 75% over a 60-second average, increase the CPU capacity by using a faster CPU, adding CPUs, or adding another machine to run the same type of DM.

Examples of DM Configurations

These examples show DM **pin.conf** file settings used with a variety of multiple CPU configurations. These examples are intended as guidelines; your settings depend on your system resources and workload.

Example 1: BRM 16-CPU database server configuration

The example depicted in Table 27-1 shows a BRM system that uses:

- A 16x450 MHz CPU database server.
- Four 6x450 MHz CPU CM/DM/EM systems.

> **Note:**
>
> The **dm_shmsize** entry is set to 64 MB to handle a larger billing load.

Table 27-1 Example 1 DM Configuration

Daemon/program	pin.conf entry	Value
dm_oracle	dm_n_fe	6
dm_oracle	dm_n_be	22
dm_oracle	dm_max_per_fe	16
dm_oracle	dm_trans_be_max	20
dm_oracle	dm_shmsize	67108864
dm_oracle	dm_bigsize	1048576

Example 2: BRM 36-CPU database server configuration

The example shown in Table 27-2 shows a BRM system that uses:

- A 36x336 MHz CPU database server.
- Four 4x400 MHz CPU CM/DM/EM systems.

Table 27-2 Example 2 DM Configuration

Daemon/program	pin.conf entry	Value
dm_oracle	dm_n_fe	4
dm_oracle	dm_n_be	24
dm_oracle	dm_max_per_fe	16
dm_oracle	dm_trans_be_max	22
dm_oracle	dm_shmsize	20971520
dm_oracle	dm_bigsize	6291456

28

Improving Interprocess Communication (IPC) Performance

Learn how to use interprocess communication (IPC) to improve performance in Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

Improving IPC Performance

By default, CM and DM processes communicate through AF_INET sockets. You can increase your system's IPC performance by configuring it to use AF_UNIX sockets between CMs and DMs that reside on the same machine and AF_INET sockets between CMs and DMs that reside on separate machines.

Both socket types are described in Table 28-1.

Table 28-1 IPC Socket Types

Socket type	Description
AF_UNIX	Provides communication through a local socket file that the DM creates each time it starts. **Note:** If the DM finds a socket file when it starts, it deletes the existing file and creates a new one. These sockets provide the fastest IPC performance but can be used only by processes located on the same machine.
AF_INET	Provides communication through an IP address. These sockets are slower than AF_UNIX sockets, but they allow communication between processes on separate machines.

To improve IPC performance by configuring your system to use both AF_UNIX and AF_INET sockets, perform these steps:

1. On machines containing both a CM and a DM, set the following entry in your CM configuration file (*BRM_home*/**sys/cm/pin.conf**):

   ```
   - cm dm_pointer 0.0.0.1 local BRM_home/sys/dm/dm_port
   ```

   If your CM also connects to DMs on other machines, such as in a multischema system, add the following entry:

> **Note:**
>
> Ensure you add a **dm_pointer** entry for each DM to which the CM connects.

```
- cm dm_pointer 0.0.0.x ip HostName PortNumber
```

where:

- *HostName* is the associated DM machine's host name or IP address.
- *PortNumber* is the associated DM's port number. The default port number is **12950**.

For an example of how to configure this file for multischema systems, see "Sample Configuration Settings for a Multischema System".

2. On machines containing both a CM and a DM, set the following entries in your DM configuration file (*BRM_home***/sys/dm/pin.conf**):

```
- dm dm_port PortNumber
- dm dm_local_socket_name BRM_home/sys/dm/dm_port
```

where *PortNumber* is the DM's port number. The default port number is **12950**.

3. On machines containing a DM but no CM, set the following entry in your DM configuration file:

```
- dm dm_port PortNumber
```

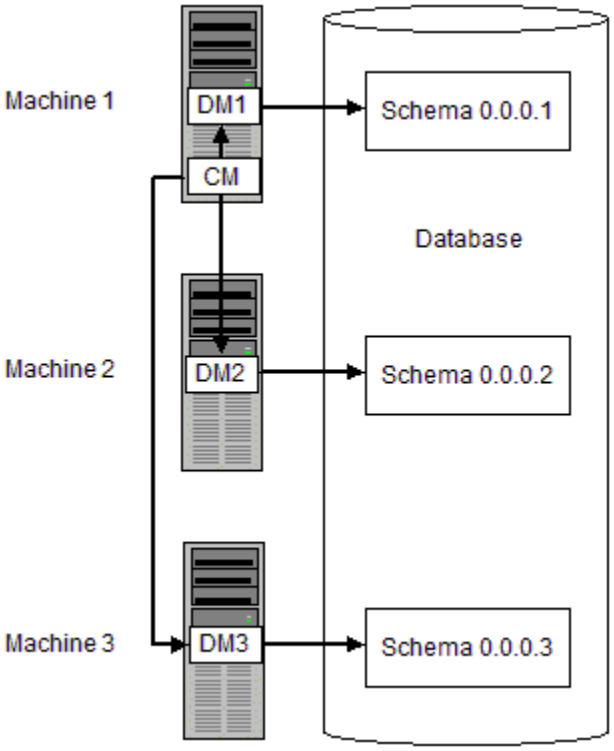
where *PortNumber* is the DM's port number. The default number is **12950**.

4. Save and close all files.

Sample Configuration Settings for a Multischema System

This section shows how to set your CM and DM configuration files for the following sample multischema system shown in Figure 28-1:

Figure 28-1 Sample Multischema System



CM pin.conf file

```
- cm dm_pointer 0.0.0.1 local BRM_home/sys/dm/dm_port
- cm dm_pointer 0.0.0.2 ip HostName2 PortNumber2
- cm dm_pointer 0.0.0.3 ip HostName3 PortNumber3
```

DM1 pin.conf

```
- dm dm_port PortNumber1
- dm dm_local_socket_name BRM_home/sys/dm/dm_port
```

DM2 pin.conf

```
- dm dm_port PortNumber2
```

DM3 pin.conf

```
- dm dm_port PortNumber3
```

29

Improving Database Performance

Learn how to improve database performance in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Improving Database Performance

A significant amount of time can be consumed in parsing SQL statements that read or write in the database. Adjusting an account, rating an event, and many other BRM activities require several steps to read and write data.

SQL statement-handle caching increases the speed at which statements are parsed.

How Statement-Handle Caching Works

An application sends an opcode through the CM to a DM, which maps each PCM operation to one or more dynamic SQL statements. For each such statement, the relational database management system (RDBMS)—such as Oracle—parses the statement, runs it, and fetches the results.

Oracle maintains a cache of the most frequent SQL queries. It uses soft parsing to shortcut its process of deciphering these statements, thus saving time in retrieving the requested data from the actual database. It then sends the data back through the DM to the application.

BRM generates and repeats a finite set of SQL statement forms. If the caching of statement handles is not enabled, the DM always parses the statement before each execution. With caching enabled, BRM maintains its most recently used statement handles within the DM, freeing the RDBMS from spending its time on soft parsing.

How to Use the Statement-Handle Cache

The **stmt_cache_entries** entry in the Oracle DM configuration file (*BRM_home***/sys/dm_oracle/pin.conf**) controls the statement-handle cache. The entry can be one of these two values:

- A value of **0** disables the cache.
- The default value of **1** means that the DM maintains 32 entries in each statement-handle cache for each back-end thread or process.

See the configuration file for more information.

The statement-handle caching performance feature requires a large value for the **open_cursors** setting in the **initSID.ora** database configuration file. See "Configuring Oracle Databases" in *BRM Installation Guide*.

> **Note:**
>
> If your Oracle database and DM both reside on computers with extraordinarily large memory resources, you might be able to cache more statement handles. Consult with Oracle for advice before attempting to cache more statement handles. It can be dangerous to exceed 32 entries because two Oracle parameters need to be increased along with **stmt_cache_entries** to prevent system failure.

Managing Database Usage

Performance of your BRM system is affected by the number of events in the database. You can limit which types of events are recorded in the database, which saves space and improves performance.

It is essential that all events with a balance impact be recorded in the database. You do this by including them in the **pin_event_map** file (*BRM_home***/sys/data/pricing/example**) when you set up your product offerings. This is enforced by the Price List Facilities Module (FM) opcodes.

On the other hand, many events without balance impacts may not need to be recorded. For example, event objects that are recorded during account creation, bundle purchase, and charge offer purchase require no further updating.

BRM provides a utility and file for excluding events from being recorded in the database. The "load_pin_event_record_map" utility loads the **pin_event_record_map** file (*BRM_home***/sys/data/config/pin_event_record_map**), in which you specify the event types to exclude.

> **Note:**
>
> - By default, if an event type is not listed in the **pin_event_record_map** file, it is recorded.
> - Event notification can still be performed based on excluded events because it is triggered even by events that are configured not to be recorded. See "Using Event Notification" in *BRM Developer's Guide*.

Events that are mapped in the **pin_event_map** file should not be added to the **pin_event_record_map** file. If you specify an event type, the event record map is ignored when the event occurs.

> **Note:**
>
> Excluding events from being recorded can cause applications that use the event data to return incorrect results. Ensure the events you exclude are not being used by any other application *before* you load the event record file.

To exclude events from being recorded:

1. Open the *BRM_home***/sys/data/config/pin_event_record_map** file.
2. List the events you want to exclude and set their record flag value to **0**. For example, to not record folds that have no balance impacts, enter:

```
/event/billing/cycle/fold:    0
```

> **Note:**
>
> The file includes the option to enable recording of listed events. You can use this option under special circumstances to record events that are normally not recorded.

3. Save and close the file.
4. Use the "load_pin_event_record_map" utility to load the file.
5. Verify that the file was loaded by using Object Browser or the **robj** command in the **testnap** utility to display the **/config/event_record_map** object. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Rebuilding Indexes

Indexes can become large and unbalanced, which reduces performance. To increase performance, rebuild the most heavily used indexes regularly. You can quickly rebuild indexes at any time. For example, you might want to rebuild some indexes before running billing. See "Rebuilding Indexes" in *BRM Installation Guide*.

Removing Unused Indexes

By default, BRM installation creates indexes for all features. However, if you do not use some features, you can delete their associated indexes.

See your database documentation for information about finding unused indexes. For example, on an Oracle database, turn on the Oracle tracing facility while BRM is running. This produces an output trace file, which you use as input to the Oracle TKPROF utility. The TKPROF utility creates a file that lists the access paths for each SQL command. These access paths include indexes.

> **Note:**
>
> You can also use the Oracle tracing facility to find missing indexes.

30
Improving Billing Performance

earn how to improve billing performance in Oracle Communications Billing and Revenue Management (BRM) by tuning billing processes or by changing business logic operations.

Topics in this document about improving the billing process:

Topics in this document about changing the business logic to improve performance:

About Billing Configuration File Entries

You can use configuration (**pin.conf**) file entries to tune the performance of billing applications. The entries in the billing configuration file are preceded with one of the following:

- **-pin_mta**: These entries set the default value for all utilities, except non-MTA utilities such as **pin_mass_refund** and **pin_recover**.
- The utility's name: These entries override the **-pin_mta** values for the specified utility.

For example, the following **children** and **fetch_size** values are used by all MTA billing utilities. The **pin_collect** and **pin_mass_refund** utilities use their own **fetch_size** values.

```
- pin_mta children 5
- pin_mta fetch_size 10000
- pin_collect fetch_size 5000
- pin_mass_refund fetch_size 8000
```

For billing performance issues not related to the configuration file, see "Additional Issues Related to Billing Performance".

Tuning the Number of Children for Billing Utilities

The **children** entry governs how many child threads will process data in parallel. Each child thread fetches and processes one account from the queue before it fetches the next account.

By default, a billing utility uses five child threads to process accounts. You can increase the number of child threads to get better billing performance when the database server remains under-utilized even though you have a large number of accounts. If you increase the number of children beyond the optimum, performance suffers from context switching. This is often indicated by higher system time with no increase in throughput.

Billing performance is best when the number of children is nearly equal to the number of DM back ends and most back ends are dedicated to processing transactions. For information on adjusting the number of DM back ends, see "Configuring DM Front Ends and Back Ends".

To tune the number of children:

1. Open the billing utilities configuration file (*BRM_home*/**apps/fm_bill/pin.conf**).
2. In the Performance Entries section, edit the **children** entry:

   ```
   - pin_mta children 5
   ```

3. Save and close the file.

Tuning the Account Cache Size for Billing Utilities (fetch_size)

The **fetch_size** entry specifies the number of account records to retrieve from the database and hold in memory before the billing utility starts processing them. In general, this value should be as large as possible to reduce the number of fetches from the database. The maximum possible fetch size depends on the complexity of the application's search results.

When running billing for parent accounts (**pay_type 10001**), the **fetch_size** value refers to the number of parent accounts to retrieve. For example, if you have 10,000 parent accounts and each account has an average of 50 children, you would set **fetch_size** to 10,000 to retrieve all of the parent accounts. If you are running billing for only the children (**pay_type 10007**), you would set **fetch_size** to 500,000 to retrieve all of the child accounts.

> **Tip:**
>
> For best performance, use a **fetch_size** value that is a multiple of the **per_batch** value.

To tune the account cache size:

1. Open the billing utilities configuration file (*BRM_home*/**apps/pin_bill_accts/pin.conf**).
2. In the Performance Entries section, edit the **fetch_size** entry. For example:

```
- pin_mta fetch_size 1000000
```

3. Save and close the file.

Tuning the Performance for the pin_collect Utility

You can tune the performance of the **pin_collect** utility by using the following configuration file entries:

- **per_batch**: This entry specifies the number of payment transactions that the **pin_collect** utility sends to **dm_fusa** in a batch. For example, if you have 20,000 payments to process and the **per_batch** entry is set to **5000**, the **pin_collect** utility would send four batches to **dm_fusa** (with each batch containing 5,000 payment transactions).
- **children**: This entry specifies how many child threads will process data in parallel. Each child thread fetches and processes one payment transaction from the queue before it fetches the next transaction.
- **fetch_size**: This entry specifies the total number of payment transactions to retrieve from the database and hold in memory before **pin_collect** starts processing them. For **pin_collect**, the optimal fetch size is the number of child threads multiplied by the number of payment transactions in a batch (**fetch_size** = **children** * **per_batch**). For example, if the **pin_collect** utility has 10 child threads and a **per_batch** size of 5,000, you would set **fetch_size** to 50,000.

To tune the performance for the **pin_collect** utility:

1. Open the billing utility configuration file (*BRM_home*/**apps/pin_billd/pin.conf**).
2. In the Performance Entries section, add **per_batch**, **children**, and **fetch_size** entries for **pin_collect**. For example:

```
- pin_collect per_batch 5000
- pin_collect children 10
- pin_collect fetch_size 50000
```

3. Save and close the file.

Filtering Search Results

Some BRM operations, such as searching for group members or searching for items to include on an invoice, can return large amounts of data and cause the Data Manager to fail. In this case, use the following configuration file entries to use a step search to find accounts in charge and discount sharing groups:

- **group_members_fetch**: Use this entry to search for members of the group sharing object when the parent group contains many members.
- **group_children_fetch**: Use this entry to search for child accounts in groups when the parent group contains many members.
- **item_fetch_size**: Use this entry when searching for items.

To filter search results:

1. Open the Connection Manager (CM) configuration file (BRM_*home*/**sys/cm/pin.conf**) in a text editor.

2. Uncomment or enter the following lines, as needed:

```
- fm_bill group_members_fetch n
- fm_bill group_children_fetch n
- fm_bill item_fetch_size n
```

   where *n* is the size of pool (memory in bytes) for the step search. The default value is 0.

3. Save and close the file.

Specifying the Number of Retries in Case of a Deadlock

For Oracle, you must specify the number of retries to be attempted in case a deadlock occurs during a billing run. For information on the deadlock error, see "Reference Guide to BRM Error Codes".

To specify the deadlock retry number:

1. Open the billing configuration file (*BRM_home***/apps/pin_billd/pin.conf)**.
2. *Uncomment* the **- pin_bill_accts deadlock_retry_count** entry.
3. Change the **deadlock_retry_count** entry, if necessary. The default is 20.

```
- pin_bill_accts  deadlock_retry_count  20
```

4. Save the file.

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

Ensuring the Sequence of Scheduled Actions

You can ensure that the **pin_deferred_act** utility processes scheduled actions for an account in the correct order by using the **group_by_account** entry. When enabled, the utility uses a single thread to process all scheduled actions for one account.

To ensure the sequence of scheduled actions:

1. Open the billing configuration file (*BRM_home***/apps/pin_billd/pin.conf)**.
2. Set the **group_by_account** entry to **1**.

```
- pin_mta  group_by_account  1
```

3. Save the file.

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

Rearranging Accounts to Improve Billing Performance

Billing utilities fetch accounts and cache them to system memory in the same sequence in which they are stored in the BRM database.

> **Note:**
>
> The number of accounts fetched from the database is determined by the **fetch_size** entry in the billing utilities configuration file.

Each account in memory is then distributed to individual child threads (or processes) for billing. This behavior may slow billing performance because of database contention.

You can sometimes improve billing performance by rearranging accounts in system memory prior to distributing the accounts to child threads for processing by using the **delta_step** entry in the billing utility configuration file.

When a value is specified for this parameter, the billing utilities rearrange accounts cached in system memory based on the parameter value specified. Generally, Oracle retrieves the accounts in the order in which they are found in the database, grouped according to their physical location on the disk.

For example, we might have 100 accounts to bill and 10 threads, as well as 10 database blocks (A, B, C, D, E, F, G, H, I, and J) that each contain 10 accounts. The database returns a list that looks like this: A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, and so on for each block. When BRM starts its threads, each of the 10 threads gets the next available account to process, and the mapping might look like this:

```
Thread 1  - A1
Thread 2  - A2
Thread 3  - A3
Thread 4  - A4
Thread 5  - A5
Thread 6  - A6
Thread 7  - A7
Thread 8  - A8
Thread 9  - A9
Thread 10 - A10
```

When a thread finishes processing an account, it takes the next available account from the list, and processing continues until all accounts have been processed. As a result, all of the threads at any given time may be accessing accounts in the same database blocks and vying for the same resources.

You can change the order in which these accounts are processed using the **delta_step** parameter. For example, to rearrange accounts by selecting and placing every tenth account cached in system memory, and then distribute these accounts to threads for billing, set this entry to **10**:

```
pin_billd delta_step 10
```

The thread mapping, instead of proceeding one account at a time, would look something like this:

```
Thread 1  - A1
Thread 2  - B1
Thread 3  - C1
Thread 4  - D1
Thread 5  - E1
Thread 6  - F1
Thread 7  - G1
Thread 8  - H1
```

```
Thread 9  - I1
Thread 10 - J1
```

The **delta_step** parameter makes it possible for each thread to be working on data from a different area of the database, reducing contention for the same resources and improving billing performance.

You can determine the optimal setting for the **delta_step** parameter by testing the billing processes and monitoring their performance. By default, this parameter is set to **0**, which means that the accounts cached in system memory are not rearranged before distribution.

To rearrange accounts in system memory:

1. Open the billing utility configuration file (*BRM_home***/apps/pin_billd/pin.conf**).
2. In the Performance Entries section, edit the **delta_step** entry. For example:

   ```
   - pin_billd delta_step 10
   ```

3. Save and close the file.

Additional Issues Related to Billing Performance

- To reduce system load, you can split large billing runs into smaller billing runs. See "Splitting a Billing Run into Multiple Runs" in *BRM Configuring and Running Billing*.
- You can improve performance of the **pin_bill_accts** utility by excluding accounts that do not need to be billed. For more information, see "About Suspending Billing of Accounts and Bills" in *BRM Configuring and Running Billing*.
- When you run the **pin_collect** utility, you can improve performance by archiving the temporary transmission logs created by the DM for the credit card processing service. See "Checking Paymentech Transmission Log Files" in *BRM Configuring and Collecting Payments* for more information about the transmission log files.
- You can create billing-related indexes just before you run billing and then delete them when billing finishes. You can add these tasks to the billing scripts. See "Running Billing Scripts" in *BRM Configuring and Running Billing* and "Removing Unused Indexes".

How the Number of Events Affects Billing

The number of events has no effect on billing or invoicing except in the following cases:

- If you defer tax calculations, billing performance is slower.
- If you create detailed invoices, performance is slower. A telephone usage invoice is an example of a detailed invoice; a fixed-fee cable subscription invoice is an example of a nondetailed invoice.

Improving Performance in Retrieving Purchased Offerings for a Bill Unit

You can improve billing performance while retrieving purchased charge offers and discount offers for a bill unit (**/billinfo** object) from the database by specifying the batch size of the number of services to search at a time.

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

To specify the batch size of the number of services to search:

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

   ```
   pin_bus_params -r Subscription bus_params_subscription.xml
   ```

3. In the file, specify the batch size of the number of services:

   ```
   <MaxServicesToSearch>5</MaxServicesToSearch>
   ```

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

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb".

Improving Performance by Skipping Previous Total Unpaid Bill for Open Item Accounting Type

You can configure BRM to skip the previous total unpaid bill for the open item accounting type when calculating the current bill. Using this business parameter, you can reduce the time taken for calculating the current bill for the open item accounting type.

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

To configure BRM to skip the previous total unpaid bill for the open item accounting type when calculating the current bill:

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

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

3. In the file, change **0** to **8**:

   ```
   <PerfAdvancedTuningSettings>8</PerfAdvancedTuningSettings>
   ```

4. Save the file as **bus_params_billing.xml**.

5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb".

Improving Trial Billing Performance by Enabling General Ledger Collection

You can configure BRM to enable the general ledger (G/L) collection for trial billing by setting the **PerfAdvancedTuningSettings** business parameter to **64**. Using this value, you can collect the G/L data required for creating trial invoices.

To enable the G/L collection for trial billing:

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

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

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. To place this file in a different directory, specify the full path name for the file. For more information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.
2. Open the **bus_params_billing.xml.out** file.
3. Search for **PerfAdvancedTuningSettings**.
4. Add **64** (0x40, set Bit 6) to the existing value of **PerfAdvancedTuningSettings**.

   For example, if the existing value is **0**, change it to **64** (0+64):

   ```
   <PerfAdvancedTuningSettings>64</PerfAdvancedTuningSettings>
   ```

   For more information on the parameter values, see the **PerfAdvancedTuningSettings** parameter description in Table 54-2.

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

5. Save the file as **bus_params_billing.xml**.
6. Go to the *BRM_home***/sys/data/config** directory which includes support files used by the **pin_bus_params** utility.
7. Use the following command to load this change into the appropriate **/config/business_params** object.

   ```
   pin_bus_param_ pathToWorkDirectory/bus_params_billing.xml
   ```

where *pathToWorkingDirectory* is the directory in which **bus_params_billing.xml** resides.

> **Note:**
>
> To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

8. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
9. Stop and restart the CM.
10. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Excluding Searches on Closed Offerings

By default, BRM retrieves active, inactive, and closed offerings during the billing process. However, most of the time, BRM does not use the data from the closed offerings.

You can configure BRM to retrieve only the active and inactive offerings by running the **pin_bus_params** utility to change the **CancelledOfferingsSearch** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To disable searches on closed offerings:

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

   ```
   pin_bus_params -r BusParamsSubsription bus_params_subscription.xml
   ```

3. In the file, change **disabled** to **enabled**:

   ```
   <CancelledOfferingsSearch>enabled</CancelledOfferingsSearch>
   ```

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

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb".

Improving Performance by Skipping Previous Total for Open Item Accounting Type When Calculating the Current Bill

> **Note:**
>
> To configure this option, you configure business profiles. For general information about business profiles, see "Creating and Managing Business Profiles" in *BRM Managing Customers*.

By default, the previous unpaid bill is calculated even for open item accounting types when calculating the current bill for a bill unit (**/billinfo** object), but the final bill does not include the previous unpaid amount. This is a time-consuming process as it involves checking all previous bill items.

To improve performance, you can configure BRM to skip calculating the previous total unpaid bill for open item accounting types.

> **Note:**
>
> To make this performance improvement for all bill units system-wide, see "Improving Performance by Skipping Previous Total Unpaid Bill for Open Item Accounting Type".

To skip the previous total for open item accounting type when calculating the current bill:

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

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

2. Set the following business profile key to **Yes**:

   ```
   <NameValue key="Skip_Prev_Total" value="Yes"/>
   ```

   where:

   - **Yes** specifies to *not calculate* the previous unpaid bill amount when calculating the current bill.
   - **No** specifies to calculate the previous unpaid bill amount when calculating the current bill. This is the default behavior when the entry is not in the file.

3. Save and close the file.

4. Run the following command, which loads the updated contents of the **pin_business_profile.xml** file into the BRM database:

   ```
   load_pin_business_profile -v pin_business_profile.xml
   ```

5. Read the object with the **testnap** utility or Object Browser and verify your changes.

For general instructions on using **testnap**, see "testnap" in *BRM Developer's Guide*.

6. Stop and restart the CM.

Improving Performance by Using Multiple Item Configurations

> **Note:**
>
> To configure this option, you configure business profiles. For general information about business profiles, see "Creating and Managing Business Profiles" in *BRM Managing Customers*.

By default, BRM uses the same item-tag-to-item-type mapping (*item configuration*) for all bill units in the system. The item configuration is used to assign bill items to events during the rating process; it also specifies which bill items are pre-created at the beginning of each billing cycle. For more information about item configuration, see "Creating Custom Bill Items" in *BRM Configuring and Running Billing*.

Using the same item configuration for all bill units can degrade performance by unnecessarily pre-creating bill items for certain types of bill units. For example, the default item configuration might pre-create cycle forward and cycle arrears items, which are normally used to track charges for postpaid bill units but are typically not required for prepaid bill units. Pre-creating such items for prepaid bill units needlessly consumes database storage space and takes more time to process during billing.

To generate fewer bill items, you can create multiple item configurations in your system and then assign the appropriate item configuration to each bill unit.

To assign an item configuration to a bill unit:

1. Create the appropriate item configuration for the bill unit.

   To create multiple item configurations, see "Setting Up BRM to Assign Custom Bill Items to Events" in *BRM Configuring and Running Billing*.

2. Open the **pin_business_profile.xml** file in an XML editor or a text editor.

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

3. In the business profile associated with the bill unit, add the following key-value pair if it does not already exist, and set the value to the name of the appropriate item configuration:

   ```
   <NameValue key="Item_Configuration" value="Item_Configuration_Name"/>
   ```

   where *Item_Configuration_Name* is the name of the item configuration to which you want to assign the bill unit. The name of the default item configuration is **Default**.

   Item configuration names are specified in the PIN_FLD_NAME field of **/config/item_tags** and **/config/item_types** objects. A pair of those objects with matching PIN_FLD_NAME values exists for each item configuration defined in your system. For more information, see "Setting Up BRM to Assign Custom Bill Items to Events" in *BRM Configuring and Running Billing*.

> **Note:**
>
> Modifying the business profile affects all bill units associated with the business profile.

4. Save and close the file.
5. Run the following command, which loads the updated contents of the **pin_business_profile.xml** file into the BRM database:

   ```
   load_pin_business_profile -v pin_business_profile.xml
   ```

6. Read the object with the **testnap** utility or Object Browser and verify your changes.

   For general instructions on using **testnap**, see "testnap" in *BRM Developer's Guide*.

7. Stop and restart the CM.

Improving Item Search Performance

If an account's paying bill unit has nonpaying child bill units, BRM performs a step search to find items. By default, BRM returns 1000 items with each step of the search.

To customize the number of items returned:

1. Open the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Change the value of the **item_search_batch** entry.

   For example, to set the number of returned items to 2000:

   ```
   - fm_bill item_search_batch 2000
   ```

   To not use step-search, set the value to **0**.

   > **Note:**
   >
   > If your paying bill unit has a large hierarchy, bypassing step-search can cause the Data Manager (DM) to run out of memory.

3. Save and close the file.

Use larger batch search memory sizes to improve run-time performance. You do not need to restart the CM to enable this entry.

Improving Performance by Skipping Billing-Time Tax Calculation

> **Note:**
>
> To configure this option, you configure business profiles. For general information about business profiles, see "Creating and Managing Business Profiles" in *BRM Managing Customers*.

In some cases, such as for prepaid accounts, taxes do not need to be calculated at billing time. If your system is configured to calculate billing-time taxes, you can improve billing performance by skipping billing-time tax calculation for individual bill units when it is not needed.

> **Note:**
>
> - To skip billing-time tax calculation for a bill unit, you configure the bill unit's business profile. Modifying the business profile affects *all* bill units associated with the business profile.
> - If a bill unit hierarchy is configured to have billing-time taxes calculated for the entire hierarchy when the paying parent bill unit is billed, do not configure the paying parent bill unit to skip billing-time tax calculation. BRM ignores the skip billing-time tax calculation setting for nonpaying bill units in such hierarchies.

To skip billing-time tax calculation for individual bill units:

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

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

2. In a business profile associated with bill units whose taxes you do not want to calculate at billing time, add the following key-value pair if it does not already exist, and set the value to **Yes**:

   ```
   <NameValue key="Skip_Deferred_Taxation" value="Yes" />
   ```

   where:

   - **Yes** specifies to *not calculate* taxes for individual bill units at billing time.
   - **No** specifies to calculate taxes for individual bill units at billing time. This is the default behavior when the entry is not in the file.

3. Save and close the file.

4. Run the following command, which loads the updated contents of the **pin_business_profile.xml** file into the BRM database:

   ```
   load_pin_business_profile -v pin_business_profile.xml
   ```

5. Read the object with the **testnap** utility or Object Browser and verify your changes.

   For general instructions on using **testnap**, see "testnap" in *BRM Developer's Guide*.

6. Stop and restart the CM.

31
Improving Invoicing Performance

Learn how to improve invoicing performance in Oracle Communications Billing and Revenue Management (BRM) by setting parameters in the invoicing configuration file.

Topics in this document:

About Tuning Invoicing Performance

Invoice utilities are multithreaded applications (MTAs) and use a similar set of configuration entries as the billing utilities, including **children**, **fetch_size**, and **per_step**. For information about these entries, see "Configuring Your Multithreaded Application" in *BRM Developer's Guide*.

Not all invoice utilities use the entries in the same way, so you can configure them individually. To specify an entry for a particular utility, replace the generic name **- pin_mta** with the name of the specific utility. For example, you might have these two entries for **fetch_size**:

```
- pin_mta          fetch_size 30000
- pin_inv_accts    fetch_size 50000
```

In this case, **pin_inv_accts** uses a fetch size of 50,000, and all other invoicing utilities use a fetch size of 30,000.

For more information, see the comments in the configuration file (*BRM_home***/apps/pin_inv/pin.conf**).

Setting the Number of Children for Invoice Utilities

The **children** entry governs how many child threads will process data in parallel. Increasing the number of child threads will provide better invoicing performance when the database server remains under-utilized even though you have a large number of accounts. If you increase the number of children beyond the optimum, performance suffers from context switching. This is often indicated by higher system time with no increase in throughput.

Because the invoice utilities work faster than the billing utilities, the number of children for invoicing can be up to 50% more than for billing or credit card processing.

You also need to tune the DM configuration file **dm_shmsize** entry to handle the number of children. A typical value for the **dm_shmsize** entry is the size of an invoice (in bytes) multiplied by the number specified in the **children** entry.

To set the number of children for invoice utilities:

1. Open the invoice utilities configuration file (*BRM_home*/**apps/pin_inv/pin.conf**).
2. In the Performance Parameters section, edit the **children** entry:

   ```
   - pin_mta children 2500
   ```

3. Save and close the file.

Tuning the Account Cache Size for Invoice Utilities (fetch_size)

The **fetch_size** entry specifies the number of invoice records to retrieve from the database and hold in memory before the invoicing utility starts processing them. In general, this value should be as large as possible to reduce the number of fetches from the database. The maximum possible fetch size depends on the complexity of the invoice utility's search results.

When running invoicing for parent accounts (**pay_type 10001**), the **fetch_size** value refers to the number of parent invoice payment accounts to retrieve. For example, if you have 10,000 parent accounts and each account has an average of 50 children, you would set **fetch_size** to 10,000 to retrieve all of the parent invoice payment accounts. If you are running billing for only the children (**pay_type 10007**), you would set **fetch_size** to 500,000 to retrieve all of the child invoice payment accounts.

If enough memory is available, set the value of the **fetch_size** entry to the number of accounts that need to be invoiced. The **fetch_size** value should be a multiple of the number specified in the **per_step** entry.

To change the account cache size for invoice utilities:

1. Open the invoice utilities configuration file (*BRM_home*/**apps/pin_inv/pin.conf**).
2. In the Performance Parameters section, edit the **fetch_size** entry. For example:

   ```
   - pin_mta fetch_size 1000000
   ```

3. Save and close the file.

Setting the Batch Size for Invoice Utilities (per_step)

The **per_step** entry specifies how much data to store in **dm_oracle** when the invoicing utility is performing a step search. It does not have a large impact on performance, but it does govern memory usage in **dm_oracle**. It also prevents BRM from using all of its memory for one large search.

A 64-bit **dm_oracle** can use reasonably large values. A typical **per_step** value for invoice utilities would be between 10,000 and 50,000.

> **Note:**
>
> When running invoicing for parent accounts (**pay_type 10001**), the **per_step** value refers to the number of parent invoice payment accounts to store in the DM. For example, if you have 10,000 parent accounts and each account has an average of 50 children, you would set **per_step** to 10,000 to store all of the parent invoice payment accounts in the DM.

To set the batch size for invoice utilities:

1. Open the invoice utilities configuration file (*BRM_home***/apps/pin_inv/pin.conf**).
2. In the Performance Parameters section, edit the **per_step** entry. For example:

   ```
   - pin_mta per_step 10000
   ```

3. Save and close the file.

Optimizing Invoicing Performance

To improve invoicing performance, you can set the **inv_perf_features** flag in the Connection Manager's configuration file (**pin.conf**) to enable or disable specific optimizations during PCM_OP_IN_MAKE_INVOICE opcode processing.

```
- fm_inv inv_perf_features 0x00000000
```

This flag is a bitmask and each bit position represents a performance optimization that can be turned on or off. By default, it is set to **0x00000000** (no optimizations are enabled).

Table 31-1 lists the bit values and their significance.

Table 31-1 Invoicing Performance Bit Values

Value	Description
0x00000001	Selects all event types. By default, only events that are configured in the **/config/invoice_events** object are selected.

Table 31-1 (Cont.) Invoicing Performance Bit Values

Value	Description
0x00000002	Selects only the following event types: • **/event/billing/adjustment/account** • **/event/billing/adjustment/event** • **/event/billing/adjustment/item** • **/event/billing/adjustment/tax_event** • **/event/billing/cycle/tax** • **/event/billing/payment/cash** • **/event/billing/payment/cc** • **/event/billing/payment/check** • **/event/billing/payment/dd** • **/event/billing/payment/payorder** • **/event/billing/payment/postalorder** • **/event/billing/payment/wtransfer** • **/event/billing/product/fee/cancel** • **/event/billing/product/fee/cycle** • **/event/billing/product/fee/cycle/cycle_arrear** • **/event/billing/product/fee/cycle/cycle_forward_bimonthly** • **/event/billing/product/fee/cycle/cycle_forward_monthly** • **/event/billing/product/fee/cycle/cycle_forward_quarterly** • **/event/billing/product/fee/cycle/cycle_forward_semiannual** • **/event/billing/product/fee/purchase** • **/event/billing/refund/cash** • **/event/billing/refund/cc**
0x00000003	• **/event/billing/refund/check** • **/event/billing/refund/dd** • **/event/billing/refund/payorder** • **/event/billing/refund/postalorder** • **/event/billing/refund/wtransfer** • **/event/billing/reversal/cc** • **/event/billing/reversal/check** • **/event/billing/reversal/dd** • **/event/billing/reversal/payorder** • **/event/billing/reversal/postalorder** • **/event/billing/reversal/wtransfer** • **/event/billing/settlement/item** • **/event/billing/writeoff/account** • **/event/billing/writeoff/bill** • **/event/billing/writeoff/item** • **/event/billing/writeoff/tax_account** • **/event/billing/writeoff/tax_bill** • **/event/billing/writeoff/tax_item** • **/event/session/dialup**
0x00000004	Creates the invoices but does not write them to the database. The bill object is updated with the invoice information.
0x00000008	Invoicing passes the input flist by reference. By default, invoicing creates a copy of the input flist. For large invoice input flists, setting this flag saves memory.
0x00000010	Events with no balance impacts are retained. If this flag is not set, events with no balance impacts are dropped from the final invoice.

Table 31-1 (Cont.) Invoicing Performance Bit Values

Value	Description
0x00000400	Specifies to keep balance impacts for sponsored events in the member accounts.

To enable multiple optimizations, you can OR the bits. For example, to select hard coded list of event types and to not write invoices to the database, set the flag to 0x00000006.

32
Improving Client Performance

Learn how to improve client performance in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Logging Noncurrency Events

By default, BRM is configured to give maximum account-creation performance by logging only events that have a balance impact associated with a currency. To log noncurrency events at the expense of system performance, change the **creation_logging** entry in the CM configuration file (**pin.conf**).

By default, these events are logged:

- **/event/billing/charge**
- **/event/billing/deal**
- **/event/billing/product**

By default, these events are not logged:

- **/event/customer/billinfo**
- **/event/customer/nameinfo**
- **/event/customer/login**
- **/event/customer/password**
- **/event/customer/status**
- **/event/billing/limit**

To log noncurrency events:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Edit the **creation_logging** entry. For example:

```
- fm_cust creation_logging 1
```

3. Save and close the file.

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

Configuring BRM to Get Events based on Balance Impact Type

You can limit the event search criteria to just one type of balance impact by changing the value of the PIN_FLD_MODE field directly or by setting the **EventChargeDiscountMode** parameter to one of the following values:

- **0:** (Default and only pre-fix mode) Returns events whose gross, net, or total balance impact matches the search criteria.
- **1:** Returns events whose gross balance impact (the sum of all charges) matches the search criteria. Discounts and tax are not considered.
- **2:** Returns events whose net balance impacts (the sum of all charges and discounts) matches the search criteria. Tax is not considered.
- **3:** Returns events whose total balance impact (the sum of all charges, discounts, and tax) matches the search criteria.

33
Improving Pricing and Rating Performance

Learn how to improve pricing and rating performance in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Changing the Precision of Rounded and Calculated Values

To improve performance, you can change the precision of rounded values and of values calculated by online rating. You change the precision by adding or modifying entries in the CM **pin.conf** file:

- To change the precision of rounded values, add or change the **rating_quantity_rounding_scale** entry. The value of this entry determines the number of digits to the right of the decimal place for rounding quantities. The default is **8**.
- To change the precision of calculated values, add or change the **rating_max_scale** entry. The value of this entry determines the number of digits to the right of the decimal place that are used. The default is **10**.

> **Note:**
>
> You must stop and restart the CM after you change these values.

Setting the Interval for Checking for Product Offering Changes

You can set the interval at which BRM checks for changes to your product offerings. If you frequently change the product offerings, you may want to use a shorter interval. If your product offerings are less volatile, you can increase the interval.

To change the interval:

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

```
- fm_rate refresh_product_interval 3600
```

The value of this entry determines the interval in seconds. The default is **3600**.

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

Setting the Interval for Updating Value Maps

To specify how frequently BRM checks for changes to value maps and updates them in the database:

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

```
- fm_zonemap_pol update_interval 3600
```

The value of this entry determines the interval in seconds. The default is **3600**.

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

Filtering the ERAs Considered during Rating and Discounting

By default, online rating checks for both account-level extended rating attributes (**/profile/acct_extrating** object) and service-level ERAs (**/profile/serv_extrating** object) when it searches for rating and discounting criteria. You can improve online rating performance by filtering the types of ERAs that BRM considers when it searches for rating and discounting criteria. For example, you can configure BRM to search for service-level ERAs only or to omit the ERA search altogether.

You can specify the types of ERAs to consider by running the **pin_bus_params** utility to change the **EnableEras** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To specify the ERA types:

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

```
pin_bus_params -r BusParamsRating bus_params_rating.xml
```

3. In the file, find this line:

```
<EnableEras>serviceAndAccount</EnableEras>
```

4. Change **serviceAndAccount** to one of the following:
   - **account**: Limits the rating and discounting criteria search to account-level ERAs by retrieving only the **/profile/acct_extrating** object.
   - **service**: Limits the rating and discounting criteria search to service-level ERAs by retrieving only the **/profile/serv_extrating** object.
   - **disabled**: Omits ERAs from the rating and discounting criteria. Because neither object is retrieved, this option provides the best performance.
5. Save the file as **bus_params_rating.xml**.

6. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_rating.xml
   ```

7. Stop and restart the CM.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Configuring the Maximum Number of Charge and Discount Offers Cached

Charge offers and discount offers that are used during the online rating process are automatically stored in the CM cache. This improves rating performance, but, over time, it can consume a large amount of memory. To prevent the CM cache from growing too large, you can set a maximum number of charge offers and discount offers that can be stored in the CM cache.

- When you set the maximum to a nonzero value, BRM prevents the online rating engine from storing more than the specified number of charge offers and discount offers in the CM cache. When the maximum number is reached, BRM flushes 10% of the charge offers and discount offers that have been used the least from the cache.

  > **Note:**
  >
  > The maximum number of charge offers and discount offers that should be stored in the CM cache depends on your business needs.

- When you set the maximum to zero, BRM does not regulate the number of charge offers and discount offers stored in the CM cache. This is the default.

To set a maximum number of charge offers and discount offers that can be cached, run the **pin_bus_params** utility to change the **ProductsDiscountsThreshold** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

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

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

3. In the file, change **0** to the maximum number of charge offers and discount offers that you would like stored in the cache. The default value of **0** specifies not to regulate the number of charge offers and discount offers in the CM cache.

   ```
   <ProductsDiscountsThreshold>0</ProductsDiscountsThreshold>
   ```

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

   ```
   pin_bus_params bus_params_rating.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Improving Performance for Loading Large Product Offerings

If you have a large product offering, you can improve performance in the following ways:

- Cache pricing data, such as G/L IDs and balance element IDs. In a test environment where you frequently modify your product offerings, caching pricing data improves performance because there is no need to load price reference objects every time you commit the product offerings to the database.

  > **Note:**
  >
  > Pricing data is created every time the CM starts. Whenever the pricing data is changed in the database, the CM must be stopped and restarted to place the new information into the cache.

  In a production system where you rarely modify your product offerings, you do not need to cache pricing data. This reserves the CM cache for other uses and eliminates the need to stop and restart the CM to update the cache if you change your product offerings.

- Turn off event logging for product offering creation events.

  > **Note:**
  >
  > When you turn off event logging, BRM still stores audit trail information for charge offers, bundles, and packages; however, an event log tracking the product offerings that were modified and who modified them is not created.

To improve loading performance:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit the **cache_references_at_start** entry. The default is **1** (cache data).

   ```
   - fm_price cache_references_at_start 1
   ```

3. Edit the **fm_price_prod_provisioning_cache** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

   ```
   - cm_cache fm_price_prod_provisioning_cache 100, 102400, 13
   ```

4. Edit the **fm_price_cache_beid** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

   ```
   - cm_cache fm_price_cache_beid 200, 524288, 32
   ```

5. Edit the **log_price_change_event** entry. The default is **0** (events are not logged).

   ```
   - fm_price log_price_change_event   0
   ```

6. Edit the **fm_offer_profile_cache** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

```
- cm_cache fm_offer_profile_cache 20, 102400, 13
```

> **Note:**
>
> For policy-driven charging, the **fm_offer_profile_cache** entry must be present in the **pin.conf** file.

7. Save the file.
8. Stop and restart the CM.

Improving Performance in Retrieving Product Details During Product Purchase

By default, BRM retrieves product details from the rating cache during product purchase. However, reading a large number of rate plans for retrieving product details can consume a lot of time. This slows down product purchase.

To speed up the product purchase process, you can configure BRM to directly retrieve the product details from the database instead of retrieving it from the rating cache during product purchase. You can do this by setting the **GetRatePlanFromCache** field in the **subscription** instance of the **/config/business_params** object to **disabled**.

> **Note:**
>
> Setting the **GetRatePlanFromCache** field to **disabled** will impact the billing performance.

To improve performance in retrieving product details during product purchase:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

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

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.
2. Search the XML file for the following line:

   ```
   <GetRatePlanFromCache>enabled</GetRatePlanFromCache>
   ```
3. Change **enabled** to **disabled**.

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

4. Save the file.
5. Change the file name from **bus_params_subscription.xml.out** to **bus_params_subscription.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility.
7. Stop and restart the CM.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

34

Improving Performance by Disabling Unused Features

Learn how to improve performance in Oracle Communications Billing and Revenue Management (BRM) by disabling unused features.

Topics in this document:

Improving Performance by Disabling Unused Features

You can improve performance by disabling features that you do not use. To do so, you edit **/config/business_params** objects. See "pin_bus_params" in *BRM Developer's Guide*.

Table 34-1 lists the features enabled by **/config/business_params** objects that most heavily impact performance.

Table 34-1 BRM Features that Heavily Impact Performance

Class	Name	Description	Default value
activity	**LightWeightAuthorization**	Enables and disables light-weight authorization.	Disabled
invoicing	**SubARItemsIncluded**	Indicates whether invoices for accounts with nonpaying child bill units include A/R items. See "Setting Defaults for Hierarchical Bill Unit Invoices" in *BRM Designing and Generating Invoices*.	Disabled
billing	**EnableARA**	Specifies whether Revenue Assurance is enabled for out-of-cycle billing. See "Enabling Billing Operations to Generate Revenue Assurance Data" in *BRM Collecting Revenue Assurance Data*.	Disabled
billing	**GeneralLedgerReporting**	Enables and disables general ledger reporting. If disabled, the journal is not generated. See "Disabling G/L Collection in BRM" in *BRM Collecting General Ledger Data*.	Enabled

Table 34-1 (Cont.) BRM Features that Heavily Impact Performance

Class	Name	Description	Default value
rating	**EnableEras**	Specifies how to enable ERAs: • **0** = No profiles • **1** = Accounts only • **2** = Services only • **3** = All See "Filtering the ERAs Considered during Rating and Discounting".	All service and account profiles
rating	**EnableGlobalChargeSharing**	Enables and disables global charge sharing. See "About Global Charge Sharing Groups" in *BRM Managing Customers*.	Disabled
multi-bal	**LockConcurrency**	Indicates the concurrency of object locking. Possible values are: • Normal (**0**): Locks the account object. • High (**1**): More concurrency of locking with greater granularity of which balance group to lock. This setting is best for performance. See "Disabling Granular Object Locking" in *BRM Developer's Guide*.	High
subscription	**GetRatePlanFromCache**	Specifies whether to retrieve rate plans from the cache during product purchase. See "Improving Performance in Retrieving Product Details During Product Purchase".	Enabled

35

Improving the Performance of Multithreaded Applications

Learn how to improve the performance of Oracle Communications Billing and Revenue Management (BRM) multithreaded applications.

Topics in this document:

Tuning Multithreaded Workloads

In most cases, increasing the number of threads that an application or process can use increases performance. However, too many threads can result in too much context switching between threads, which can decrease performance.

To determine the optimum number of threads, increase the number of threads and watch the CPU utilization. If adding threads increases system time, adding threads is not helping performance.

Controlling Thread Load on Multithreaded Applications

The number of child threads allowed is controlled by the **children** entry in the application's **pin.conf** file. You can set the number of threads manually by editing the **pin.conf** file or automatically by using the **pin_mta_monitor** monitoring utility.

To improve performance, you can limit the number of child threads. For example, you can limit the number of threads during the business day to prevent excessive load on the network and increase the number of threads during off-peak hours.

To control the number of threads for an MTA by using **pin_mta_monitor**:

> **Note:**
>
> After entering a command at the monitor prompt, press the spacebar once and then press Enter. If you do not enter a space after the command, the utility does not run the command.

1. Go to the directory from which you run the MTA.
2. Start the **pin_mta_monitor** utility by running the following command:

   ```
   pin_mta_monitor mta_application
   ```

where *mta_application* is the MTA that **pin_mta_monitor** tracks.

3. The **(mon)>** prompt appears. Provide the number by which you want to increase or decrease the number of threads that *mta_application* uses (press the spacebar and then press Enter).
   - To increase the number of threads that *mta_application* uses, enter:

     ```
     (mon)> t+number
     ```

   - To decrease the number of threads that *mta_application* uses, enter:

     ```
     (mon)> t-number
     ```

For example, the following commands increase the thread pool of **pin_inv_export** by two threads:

```
pin_mta_monitor pin_inv_export
(mon)> t+2
```

Monitoring the Thread Activity of Multithreaded Applications

Use the **pin_mta_monitor** utility to monitor the thread activity of MTAs.

> **Note:**
>
> After entering a command at the monitor prompt, press the spacebar once and then press Enter. If you do not enter a space after the command, the utility does not run the command.

To monitor the thread activity of an MTA:

1. Go to the directory from which you run the MTA.
2. Start the **pin_mta_monitor** utility by running the following command:

   ```
   pin_mta_monitor mta_application
   ```

   where *mta_application* is the MTA that **pin_mta_monitor** tracks. For example, the following command is used to monitor the activity of **pin_inv_export**:

   ```
   pin_mta_monitor pin_inv_export
   ```

3. The **(mon)>** prompt appears.

   To print the thread activity of the *mta_application* to **stdout**:

   ```
   (mon)> p
   ```

   The following is a sample output of the thread activity that **pin_mta_monitor** prints for **pin_inv_export**:

   ```
   Required:2, cmd:print
   Thread_id:2, Working POID:0.0.0.1 /invoice 221084 0, Flag:0
   Thread_id:3, Working POID:0.0.0.1 /invoice 220860 0, Flag:0
   ```

4. To stop printing the thread activity of the *mta_application* to **stdout**:

   ```
   (mon)> q
   ```

36
Troubleshooting Performance

Learn how to troubleshoot performance problems in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Troubleshooting Poor Performance

When troubleshooting poor performance, first consider the following:

- Under-configured hardware.
- Inefficient table layout.
- Database bottlenecks.
- Inefficient custom application code.
- Repeated runtime errors resulting from configuration problems.

In addition, you can look for different problems depending on whether CPU utilization is high or low.

Low Performance with High CPU Utilization

If performance is low and CPU utilization is high, or if there are performance spikes, there is probably a configuration or indexing issue. Check the following:

- Hardware limitations.
- Table/volume layout.
- Spin count is too high.
- Lack of proper indexes. This can show up as very high CPU utilization with no other apparent problems except for a high number of processes. Find which columns are being accessed in the operation being performed and ensure that they are properly indexed.
- Not enough database buffers.
- Swapping.
- Kernel parameters too low.

Low Performance with Low CPU Utilization

If performance is low and CPU utilization is low, check for a bottleneck between different system tiers (for example, between the DM and the database).

- Use the database monitoring tools to analyze the performance of the database system.
- Use SQL tracing and timing to check for inefficient application code.
- Check for an under-configured BRM system, which could be one of the following:
  - CM Proxy with a low number of children.
  - DMs with a low number of back ends.
  - System logging level is too high.

Monitor the DM system utilization and Oracle system utilization and tune the number of DM back ends accordingly. A good starting point for DM back-end numbers is eight times the number of processors.

For more information, see "Improving Data Manager and Queue Manager Performance".

Quick Troubleshooting Steps

- Run quick timing tests by using the **testnap** utility with **op_timing** turned on to ping each CM and DM (with the PCM_OP_TEST_LOOPBACK opcode). If the operations are relatively slow, it indicates a problem in the basic configuration.
- Run the system with a log level of DEBUG on the CM and DM and analyze log files.
- Check for network collisions and usage data.
- Check if you have logging (debugging) turned on in the CM. Logging is good for troubleshooting, but it should not be turned on in a production environment because it reduces performance.
- Performance parameters in **pin.conf** files should be large enough to handle the load. The most likely problems are in the DM entries.
- Check if you have enough DM back ends to handle your transaction load.
- Try putting tables and indexes on different disks.
- Check the size of redo and rollback logs and database configuration parameters.
- Send a few **kill -USR1** commands to the DMs and CMs that seem to be having problems. This causes them to dump their state to the BRM error log files. Snapshots should be up to 20 minutes apart. These log files may contain information that indicates the nature of the problem.
- Turn on SQL tracing and analyze query plans. Look for full table scans. Ensure that indexes are on the appropriate columns for the query being run. Especially verify for any customizations.
- Turn on the **timed_statistics** parameter. Look for unusually long execution times for SQL commands.
- Monitor hardware activity:

  - On Linux, and Solaris systems, use **vmstat**, **netstat**, and **sar**.
  - Drill down to the storage device level by using **sar** with the **-d** parameter. This should help you find the source of the problem.

> **Note:**
>
> If the file systems are configured from logical volumes that are comprised of physical disks, different file systems could be sharing the same underlying disk. It is important to unravel who owns what in order to isolate potential contention (waiting on I/O).

- Problems such as intermittent daemon failures can be indicated by core files. Try the following command to locate them:

```
% find BRM_home -name core -exec file {} \;
```

  If there are no core files, try turning on maximal debugging. You do not want to do this for very long, especially on a production system, because the log files fill up rapidly.

```
% pin_ctl stop cm
% setenv CMAP_DEBUG to 0x1331f3
% setenv CM_DEBUG to 0x0001
% setenv cm_loglevel to 3
% pin_ctl start cm
```

  System level tracing can also be useful:

```
# ps -ef | grep cm
# truss -p cm_pid
```

Part IV
Troubleshooting BRM

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

- Resolving Problems in Your BRM System
- Diagnosing Some Common Problems with BRM
- Using Error Logs to Troubleshoot BRM
- Reference Guide to BRM Error Codes

37
Resolving Problems in Your BRM System

Learn how to troubleshoot your Oracle Communications Billing and Revenue Management (BRM) system.

Topics in this document:

- General Checklist for Resolving Problems with BRM
- Contacting Technical Support

See also "Diagnosing Some Common Problems with BRM", "Using Error Logs to Troubleshoot BRM", and "Reference Guide to BRM Error Codes".

General Checklist for Resolving Problems with BRM

When any problems occur, it is best to do some troubleshooting before you contact Oracle Technical Support:

- You know your installation better than Oracle Technical Support does. You know if anything in the system has been changed, so you are more likely to know where to look first.
- Troubleshooting skills are important. Relying on Technical Support to research and solve all of your problems prevents you from being in full control of your system.

If you have a problem with your BRM system, ask yourself these questions first, because Oracle Technical Support will ask them of you:

- What exactly is the problem? Can you isolate it? For example, if an account causes a BRM component to crash on one computer, does it give the same result on another computer? Or if users cannot authenticate, is it all services or just one service? If it is an IP problem, does it affect all users or just those on a specific POP or a specific type of terminal server?

  Oracle Technical Support needs a clear and concise description of the problem, including when it began to occur.
- What do the log files say?

  This is the first thing that Oracle Technical Support asks for. Check the error log for the BRM component you are having problems with. If you are having problems connecting to BRM, start with the log for the CM.

  See "Using Error Logs to Troubleshoot BRM".
- Have you read the documentation?

  Look through the list of common problems and their solutions in "Diagnosing Some Common Problems with BRM".
- Has anything changed in the system? Did you install any new hardware or new software? Did the network change in any way? Does the problem resemble another one you had previously? Has your system usage recently jumped significantly?

- Is the system otherwise operating normally? Has response time or the level of system resources changed? Are users complaining about additional or different problems?
- If the system is completely dead, check the basics: Can you run **testnap** successfully? Can you access Oracle outside of BRM? Are any other processes on this hardware functioning normally?

If the error message points to a configuration problem, check the configuration file (**pin.conf** or **properties**) for the troublesome component. If you find that the solution requires reconfiguring the component, stop all processes for that component, change the configuration, and stop and restart the component.

For more information, see:

- Using Configuration Files
- Starting and Stopping the BRM System

If you still cannot resolve the problem, contact Oracle Technical Support.

Contacting Technical Support

If you cannot resolve your problems with BRM, contact Oracle Technical Support.

Problems can often be fixed simply by shutting down BRM and restarting the computer that the BRM system runs on.

If that does not solve the problem, the first troubleshooting step is to look at the error log for the application or process that reported the problem. See "Using Error Logs to Troubleshoot BRM". Be sure to observe "General Checklist for Resolving Problems with BRM" before reporting the problem to Oracle Technical Support.

If "General Checklist for Resolving Problems with BRM" does not help you resolve the problem, write down the pertinent information:

- A clear and concise description of the problem, including when it began to occur.
- Relevant portions of the relevant log files.
- Relevant configuration files (**pin.conf** or **properties**).
- Recent changes in your system, even if you do not think they are relevant.
- List of all BRM components, ServicePaks, FeaturePaks, and patches installed on your system.

> **Note:**
>
> You can collect BRM information by using the **pinrev** script. For information, see "Checking BRM Component Version Numbers".

When you are ready, report the problem to Oracle Technical Support.

38

Diagnosing Some Common Problems with BRM

Learn about some common problems that you might find when troubleshooting Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Problems Starting BRM Components

Problem: Bad Bind, Error 13

One of the log files (**log** or **pinlog**) for the DM or CM has a reference to “bad bind" and “errno 13".

Cause

The port number specified in the configuration file (**dm_port** or **cm_ptr** entry) is incorrect.

Another possibility is that the port number is below 1023, and the CM, CMMP, or DM was not started as **root**. System processes that use port numbers below 1023 must be started as **root**. If you use a port number greater than 1024, you do not have to start the process as **root**.

Solution

Edit the configuration file for the component to specify an unassigned port number above 1023, such as 1950.

Problem: Bad Bind, Error 125

The log file for the DM or CM has a reference to “bad bind" and “errno 125".

Cause

Duplicate port number. Some other process is already using the port.

Solution

Edit the configuration file for the component to specify an unassigned port number above 1023, such as 1950.

Problem: Cannot Connect to Oracle Database

When you look at the processes running, you see the Oracle DM master and front ends running but no back end running.

Causes

- The database name configuration entry (**sm_database**) for the DM points to the wrong database name. (The error message shows which database name the DM is trying to connect to.)
- The Oracle password configuration entry (**sm_pw**) is missing.
- The Oracle **tnsnames** file is missing or incorrect.
- The **oracle_sid** or **oracle_home** environment variable is set incorrectly.
- The Oracle DM is spawning too many back-end processes simultaneously for the IPC or BEQ protocol to handle.

Solutions

- Enter the correct database name and Oracle user name and password for the BRM database in the configuration file for the Oracle DM and restart the DM.
- Create a valid Oracle **tnsnames** file and check the environment variables.
- If you are using the IPC or BEQ protocol, configure the Oracle DM to wait a specified amount of time before spawning or respawning a new back-end process. To do this, add the following entry to the Oracle DM configuration file (*BRM_home***/sys/dm_oracle/pin.conf**), where *BRM_home* is the directory in which BRM is installed:

  ```
  - dm dm_restart_delay DelayTime
  ```

  > **Note:**
  >
  > Adding a delay increases the Oracle DM startup time.

  where *DelayTime* is the amount of time, in microseconds, the Oracle DM should wait before spawning a new back-end process. Set *DelayTime* to the smallest possible time that fixes your connection problems. As a guideline, start with 1000000 microseconds (1 second) and then decrease the time until you find the optimal setting for your system.

Problem: Lost TCP Connections

Cause

BRM recognizes when an application closes a TCP connection. If the computer running the client application fails, however, the application might not close the TCP socket.

Solutions

In the **pin.conf** files for the CM and the Connection Manager Master Process (CMMP), the **keepalive** entry specifies whether to monitor the TCP connection. See "Enabling Java PCM Clients to Use Operating System TCP/IP Keepalive Parameters".

> **Note:**
>
> This entry should be set to avoid sockets not being closed properly due to network problems or hardware crashes.

The CM monitors the TCP connections by using the standard TCP **keepalive** feature. This lets you detect lost connections and clean up the CM and DM.

With the **keepalive** feature turned on, BRM uses the system's **keepalive** APIs to detect a lost connection and to try to reconnect, before closing the socket.

For more information on TCP **keepalive** options, see the TCP and **keepalive** documentation for your operating system.

Problem: Hung and or Looping Processes

A *hung* process does not respond in a normal fashion.

A *looping* process uses CPU cycles without doing any useful work.

Cause

If the CM does not respond to a login attempt, one of the processes in the system might be hung.

If the CPU time for a process is increasing and is out of proportion to the rest of the processes, this might be a looping process.

Solutions

Check the status of the CM. See "Monitoring Connection Manager Activity". The CM should show a new connection. If the CM report shows that the CM is “waiting on DM," the DM might be hung. You can check the database by verifying that it responds to manual SQL commands.

To check the CPU time used by a process, enter the following command twice, separated by a 10- to 30-second interval (or as much as several minutes on a lightly loaded system):

```
ps -ef | grep process
```

Stopping a Hung or Looping Process

> **Note:**
>
> Before you stop a hung or looping DM or CM process, check its status at least twice at 30-second intervals (or up to several minutes on a lightly loaded system). For more information, see "Monitoring Data Manager Activity" or "Monitoring Connection Manager Activity".

Enter the following command to stop a hung or looping process:

```
kill -ABRT process_id
```

BRM stops the process and writes a **core** image file of the process. If you contact Oracle Technical Support about this problem, send the **core** file along with the relevant log files. (See "Contacting Technical Support".)

Problem: ORA-01502: Index 'PINPAP.I_EVENT_ITEM_OBJ__ID' or Partition of Such Index Is in Unusable State

While loading the CDRs using the direct path load option, an error stating that the index is in an unusable state occurs.

Cause

While IREL processes the CDRs using the direct path loading option, it updates the indexes. However, as the index is being updated, another application, for example, **pin_monitor_balance** would also access the same index partition.

Solution

Configure the **dm_sql_retry** entry in the **pin.conf** file. This is specified as an integer value that indicates the number of times an SQL statement is to be retried if this error occurs.

> **Note:**
>
> This is not a mandatory parameter to be set in the **pin.conf** file. The default behavior is to not try running the SQL statement if the error occurs.

Problems Stopping BRM Components

- Problem: No Permission to Stop the Component
- Problem: No pid File

Problem: No Permission to Stop the Component

You run the stop script, but the script fails. You find a reference to “permission denied" in the log file for the component.

Cause

You do not have permission to stop the BRM system.

Solution

Log in as **root** or as the user who started the BRM system.

Problem: No pid File

You run the stop script, but the script fails. You find a reference to “no pid file" in the log file for the component.

Cause

BRM cannot find the **.pid** file.

Solution

Identify the process ID for the component you want to stop, and then stop the process manually. See "Starting and Stopping the BRM System".

Problems Connecting to BRM

- Problem: Cannot Connect to the Database
- Problem: AMM Takes Longer Time to Connect to the Database
- Problem: Cannot Connect to the CM
- Problem: CM Cannot Connect to a DM

Problem: Cannot Connect to the Database

When you try to start a client application, you get an error message advising you of “problems connecting to the database."

Cause

The CM might not be set to handle the number of current client sessions.

Solution

Set the **cm_max_connects** entry in the configuration file for the CM to a number larger than the number of client sessions you anticipate. Then restart the CM.

Problem: AMM Takes Longer Time to Connect to the Database

When you try to enable the Account Migration Manager (AMM) jobs by running the **pin_amt** utility, the Authentication lapse error is logged in the **pin_amt.log** file.

Cause

When enabling the migration jobs, AMM takes much longer time than expected to connect to the Oracle database.

Solution

Do the following:

1. Open the *BRM_home*/**bin/pin_amt** file in a text editor.
2. Search for the following entries in the file:

```
$JAVA -DPIN_HOME=$PIN_DIR
-DJAVA_OPTS="-Xmx2048m -Xms256m" com.portal.amt.AmtUI $*
```

3. Replace the entries with the following:

```
$JAVA -Djava.security.egd=file:/dev/./urandom
-Dsecurerandom.source=file:/dev/./urandom
-DPIN_HOME=$PIN_DIR -DJAVA_OPTS="-Xmx2048m -Xms256m" com.portal.amt.AmtUI $*
```

4. Save and close the file.

Problem: Cannot Connect to the CM

An application cannot connect to BRM, and the log file for the application (which might be **default.pinlog** in the current directory) shows the error "PIN_ERR_NAP_CONNECT_FAILED(27)."

Causes

- The configuration file (**pin.conf**) for the application might be pointing to the wrong CM.
- The CM is not running.
- The CM is not set to handle this many connections.
- No TCP sockets are available on the client or CM machine, perhaps because you used many sockets recently and the sockets have not been released from their two-minute wait period after the connections were closed.

Solutions

- Open the configuration file for the application and check the entries that specify the CM.
- Check for CM processes. See "Checking the Number and ID of a BRM Process".
- Set the **cm_max_connects** entry in the configuration file for the CM to a number larger than the number of application sessions you anticipate. Then restart the CM.
- Wait a few minutes to see if the sockets are freed up.

  On Solaris: To see how many sockets are available:

  ```
  netstat -n -f inet -p tcp | wc -l
  ```

  On Linux: To see how many sockets are available:

  ```
  netstat -n -A inet -t | wc -l
  ```

  If the resulting number is close to 65535, there are too many socket connections for a single IP address on this machine.

Problem: CM Cannot Connect to a DM

You might find a message similar to the following:

```
DMfe #3: dropped connect from 111.122.123.1:45826, too full
W Thu Aug 06 13:58:05 2001 portalhost dm:17446 dm_front.c(1.47):1498
```

Cause

There are not enough connections allowed for the DM.

Solution

- Use the **dm_max_per_fe** parameter in the DM configuration file to increase the number of CM connections allowed.
- Install and configure an additional DM.

Problem: Rated Event Loader Cannot Connect to the CM

If SSL is enabled in BRM, sometimes Rated Event Loader fails to connect to the CM with the following error message:

```
An error occurred while attempting to connect to the Infranet CM.
Please validate the infranet.connection property value and ensure the CM is
running.
```

Cause

There might be a heavy load on Rated Event Loader.

Solution

Do the following:

1. Create a copy of the *BRM_home*/**wallet/client** directory by running the following command:

   cp -r *BRM_home*/**wallet/client** *BRM_home*/**wallet/client_ssl**

2. Open the *BRM_home*/**apps/pin_rel/Infranet.properties** file in a text editor.
3. Modify the **infranet.pcp.ssl.wallet.location** entry to point to the absolute path of the directory that you created in step 1:

   infranet.pcp.ssl.wallet.location = *BRM_home*/**wallet/client_ssl**

Problems with Deadlocking

Problem: BRM “Hangs" or Oracle Deadlocks

Your BRM system stops responding, or Oracle reports deadlocking messages.

Cause

The DM might have too few back ends for the type of BRM activity.

Solution

Configure the DM with more back ends. For example, provide at least two DM back ends for each customer service representative. For more guidelines on setting the number of back ends, see "Improving Data Manager and Queue Manager Performance".

Problem: dm_oracle Cannot Connect to the Oracle Database

The Oracle DM (**dm_oracle**) waits indefinitely for a response from the Oracle database.

Cause

If there is a problem with the Oracle database, **dm_oracle** might stop responding when it attempts to connect to the database.

Solution

Set the **database_request_timeout_duration** parameter in the **dm_oracle** configuration file (*BRM_home*/**sys/dm_oracle/pin.conf**):

```
- dm database_request_timeout_duration milliseconds
```

where *milliseconds* is the number of milliseconds the DM waits for a response. For example:

```
- dm database_request_timeout_duration 10000
```

If the database does not respond during the wait period and you are using Oracle RAC, the DM times out and then makes one attempt to connect to another Oracle database instance.

If this **pin.conf** parameter is not specified or is set to **0**, the connection attempt does not time out.

> **Note:**
>
> - If you are using a single database or a multischema system without Oracle RAC, the DM attempts to connect to the same database schema again. In this case, the *timeout* setting is useful only if you are experiencing temporary network problems.
> - If you are using Oracle RAC, the **tnsnames.ora** file must be configured correctly for the reconnection to work.

Problems with Memory Management

Problem: Out of Memory

The DM will not start, and the error log file for the DM refers to “bad shmget" or “bad shmat" and “errno 12." Or, when the system is running, the CM or an application shows the error “PIN_ERR_NO_MEM" in its log file.

Causes

The DM or another queuing-based daemon did not have enough shared memory to complete the operation. This is caused by one or more of the following conditions:

- Other processes are using all of the shared memory.
- There are too many CM processes.
- There are memory leaks in the CM or its FMs.
- **On Solaris:** The shared memory segment allocated by one of the DM processes has not been cleaned up properly, leaving a sizeable chunk of memory allocated but unused. This condition, rare in normal operation, can be caused by the following activities:
  - Repeated starting and stopping of the system.
  - Stopping the DMs manually, especially by using **kill -9**.
- **On Solaris:** The shared memory configuration for the system is less than the shared memory set for BRM.

Solutions

To check for memory leaks, use **ps** with the **vsz** flag at two or more intervals to see changes in shared memory.

On Solaris:

```
ps -eo pid,vsz,f,s,osz,pmem,comm | egrep 'cmldml [application]'
```

On Linux:

```
ps -eo pid,vsz,f,s,sz,pmem,comm | egrep 'cmldml [application]'
```

For a CM, **vsz** should grow only until an operation has passed through the CM and then stay constant. For example, if **vsz** is growing during billing, there is a memory leak.

To check for and clean up unused memory on Solaris:

1. Stop all DM processes. See "Starting and Stopping the BRM System".
2. Confirm that there are no DM processes running. See "Checking the Number and ID of a BRM Process".
3. Run **df -k** to check swap space usage. Confirm that the available space is very low.
4. Run **ipcs -ma** to show the shared memory segments that have been allocated but not used recently. A shared memory segment is probably abandoned when you see the following conditions:
   - Number of attaches (NATTCH) is 0
   - KEY is 0 (and not using a special **dm_shmkey**)
   - Creator process ID (CPID) is gone
   - Last detach time (DTIME) has a value

5. Run **ipcrm -m** *segment_id* on each of the unused segments to free up the space.
6. Run **df -k** again to confirm that the available swap space has been cleared.
7. Stop and restart the DM processes.

To increase the system shared memory on Solaris, open the **/etc/system** file and set the **shminfo_shmmax** configuration parameter to a value greater than the value of **dm_shmsize** in the DM configuration file (**pin.conf**). Stop and restart the computer.

Example **/etc/system** file for a 64 MB system:

```
set shmsys:shminfo_shmmax=37748736
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
set semsys:seminfo_semmns=200
set semsys:seminfo_semmni=70
```

In this example, the shared memory segment has been set to 36 MB (1048576 times 36).

To check for and clean up unused memory on Linux:

1. Stop all DM processes. See "Starting and Stopping the BRM System".
2. Confirm that there are no DM processes running. See "Checking the Number and ID of a BRM Process".
3. Run **df -k** to check swap space usage. Confirm that the available space is very low.
4. Run **ipcs -ma** to show the shared memory segments that have been allocated but not used recently.
5. Run **ipcs -mac** to show the shared memory segments that have been allocated along with the corresponding user information.
6. Run **ipcs -mat** to show the shared memory segments that have been allocated detach timing information.

> **Note:**
>
> In steps 4, 5, and 6, a shared memory segment is probably abandoned when you see the following conditions:
>
> - Number of attaches (NATTCH) is 0
> - KEY is 0 (and not using a special **dm_shmkey**)
> - Creator process ID (CPID) is gone
> - Last detach time (DTIME) has a value

7. Run **ipcrm -m** *segment_id* on each of the unused segments to free up the space.
8. Run **df -k** again to confirm that the available swap space has been cleared.
9. Stop and restart the DM processes.

To increase the system shared memory on Linux, open the **/etc/sysctl.conf** file and set the **shminfo_shmmax** configuration parameter to a value greater than the value of **dm_shmsize** in the DM configuration file (**pin.conf**). Stop and restart the computer.

Problem: Java Out of Memory Error

When using GUI applications such as Developer Center or batch applications such as Invoice formatter, you might sometimes receive “java.lang.OutOfMemoryError: Java heap space" error messages.

Cause

The Java application does not have enough memory to complete the operation.

Solution

Increase the maximum heap size used by the Java Virtual Machine (JVM). The exact amount varies greatly with your needs and system resources.

The heap size is controlled by the **-Xmx** size entry in the Java application startup script. By default, the **-Xmx** size entry is not present in the startup line. To increase the maximum heap size, add this entry and a number (in megabytes) to the application startup line. The following example adds a 1024 MB maximum heap size to the class:

```
java -Xmx1024m class
```

> **Note:**
>
> Increasing the heap size may degrade the performance of other processes if insufficient resources are available. You must adjust the heap size based on your application needs and within your system's limits.

Problem: Memory Problems with the Oracle DM

The error log file for the DM for your Oracle database refers to “No memory for...", such as “No memory for list in pini_flist_grow." You suspect memory problems, but your system has sufficient memory for the environment.

Cause

The DM is not configured to use sufficient shared memory.

Solution

1. Open the DM configuration file (*BRM_home*/**sys/dm_oracle/pin.conf**).
2. Increase the size of the **dm_bigsize** and **dm_shmsize** parameters. Follow the guidelines in the configuration file for editing these entries.
3. Save the configuration file.
4. Stop and restart the DM.

Problems Running Billing

Problem: Billing Daemons Are Running, but Nothing Happens

Even though the billing processes are running, BRM is not producing billing data.

Cause

There are too few back ends for the DM. Because billing daemons run in parallel, you must have at least one DM back end for each billing program thread, plus one back end for the main thread searches.

Solution

Edit the **dm_n_be** entry in the DM configuration file (**pin.conf**) to add more back ends to the DM, and then stop and restart the DM. See "Configuring DM Front Ends and Back Ends".

Problem: High CPU Usage for the Number of Accounts Processed

Running the billing scripts puts an inordinately heavy load on the computer, and processing the accounts takes a long time.

Cause

An index is missing or unbalanced; or in Oracle, an index is in the CHOOSE Optimizer mode and statistics are out of date.

Solution

Rebuild the BRM indexes before you run the billing scripts. See "Rebuilding Indexes" in *BRM Installation Guide*.

Problems Creating Accounts

Problem: fm_delivery_mail_sendmsgs Error Reported in the CM Log File

Cause

BRM is trying to send a welcome email message, but the Email DM (**dm_email**) is not running.

Solution

Start the Email DM, or disable the welcome email message.

- To start the Email DM, see "Sending Email to Customers Automatically" in *BRM Managing Customers*.
- To disable the welcome message, see "Setting up Welcome Messages to Customers" in *BRM Managing Customers*.

Problems Loading Configuration Objects

Problem: Failed to create XML context in isXsltExists, error [266]

Cause

The **load_config** utility tried to load the contents of XML configuration files into configuration (/config/*) objects in the BRM database, but the contents are not loaded.

Solution

Set the ORACLE_HOME environment variable to the BRM database client library path; for example, /tools/CGBU/contrib/Linux/x86_64/packages/oracle/db/12.2.0.1.0.

Problems During BRM Upgrade

Problem: The BRM root passwords stored in the wallet and /service/pcm object are not matching

A validation error appears due to incorrect BRM root password during the BRM installation for upgrade.

Cause

You have provided an incorrect BRM root password during the BRM installation for upgrade.

Solution

Change the incorrect BRM root password in the BRM client wallet by using the **pin_config_editor** utility. See "Changing Incorrect BRM Root Password" for more information.

> **Note:**
>
> You can also change the BRM root password in the BRM client wallet by using the PCM_OP_CUST_UPDATE_SERVICES opcode. For more information on the opcode, see *BRM Opcode Guide*.

Changing Incorrect BRM Root Password

To change the incorrect BRM root password:

1. Go to the *BRM_home***/bin** directory.
2. Run the following command:

```
pin_config_editor -setconf -wallet clientWalletLocation -parameter
-.login_pw -pwd
```

where *clientWalletLocation* is the path to the BRM client wallet.

3. At the command prompt, enter the existing BRM root password.

> **Note:**
>
> Ensure that you provide only the existing BRM root password. You can retrieve the existing BRM root password by using **pin_crypt_app** utility. See "Retrieving Configuration Entries from Client Wallet for Java PCM Applications" for more information.

4. Enter the wallet password.
5. Run the following command:

```
pin_config_editor -setconf -wallet clientWalletLocation -parameter
infranet.connection
```

6. At the command prompt, enter values listed in Table 38-1.

Table 38-1 BRM Connection Information

Field	Description
User Name	The user name for connecting to BRM.
Password	The BRM user's password.
Host Name	The IP address or the host name of the machine on which the primary BRM Connection Manager (CM) or CM Master Process (CMMP) are running.
Port Number	The TCP port number of the CM or CMMP on the host computer. The default value is **11960**.
Service Type	The BRM service type. The default value is **/service/admin_client**.
Service POID Id	The POID of the BRM service. The default value is **1**.
Wallet Password	The password for the BRM wallet.

39
Using Error Logs to Troubleshoot BRM

Learn how to read error messages when troubleshooting Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Using Error Logs to Troubleshoot BRM

BRM error log files provide detailed information about system problems. If you are having a problem with BRM, look in the log files.

Log files include errors that need to be managed, as well as errors that do not need immediate attention (for example, invalid logins). To manage log files, you should make a list of the important errors for your system, as opposed to errors that do not need immediate attention.

Understanding Error-Message Syntax

BRM error messages use this syntax:

```
[severity] [date_&_time] [host_name] [program]:[pid] [file]:[line] [correlation_id]
[message]
```

where:

- *severity* is the severity of the error message. It can be one of the following values:
  - **E:** Error. An error indicates that a component of your BRM system is not operating correctly. This is the most severe type of problem.
  - **W:** Warning. Warnings indicate that the system cannot perform a task because the database contains incorrect or inconsistent data. The system continues to operate, but you should investigate and resolve problems with the data immediately.
  - **D:** Debug. Debugging messages, which indicate problems with an application, are typically used by application developers to diagnose errors in custom applications. You see these messages only if you set error reporting to the highest level.
- *date_&_time* is the date and time the message was logged.
- *host_name* is the name of the computer generating the message. If several machines are sharing a log file using Network File System (NFS), or if all log files are stored in a central location, use this information to pinpoint the machine with the problem.

- *program* is the name of the program (or process) generating the log message. This information helps resolve problems in billing utilities, for example, because all billing utilities use the same log file.
- *pid* is the process ID for the process generating the log message. This information helps resolve problems in components, such as Connection Managers (CMs) and Data Managers (DMs), that might have many processes running in parallel with the same name (*program*).
- *file* is the name of the source file where the error was detected. Technical Support uses this information when diagnosing system problems.
- *line* is the line number in the source file where the error was detected. Technical Support uses this information when diagnosing system problems.
- *correlation_id* is the identifier for all error messages related to a single error occurrence. This information can be used to sort error messages and to identify the set of error messages generated from a single error occurrence.
- *message* is a detailed description of the error condition. Part of the message often includes the error type, location, and code, which you can use to interpret the error. See "Reference Guide to BRM Error Codes".

Resolving Clusters of Error Messages

An error often produces a cluster of error messages in the log file. The Facilities Modules (FMs), especially, tend to generate cascading messages. To resolve the error, isolate the group of messages, as defined by their common *correlation ID*, date/time, and process ID, and look at the first one in the series. The error location for that message generally indicates the source of the problem. Then find the last message text in the first error, to identify the operation that was associated with the error. Always consider whether an error could have been caused by something happening in a downstream process.

Logging External User Information in Error Logs

When using an external application to connect to BRM, if a request from the external application fails, BRM logs the BRM user, the external user and the external correlation ID in the *correlation_ID* of the log header, to identify the original request from the external application.

The additional information in the *correlation_ID* uses the following syntax:

```
BRM_user::external_user:external_correlation_ID
```

Where:

- *BRM_user*: Specifies the user with which BRM client connects to the CM.
- *external_user*: Specifies the user from the external system connecting to BRM.
- *external_correlation_ID*: Specifies the correlation ID that an external system passes to BRM in order to correlate between an operation within its system and the corresponding operations in BRM.

For example:

```
2:CT1255:Account_Manager:1948:1684:63:1063403309:14:root.0.0.0.1::user1:123456789
```

> **Note:**
>
> If the external application does not provide the external user and external correlation ID, the *correlation_ID* displays empty strings.
>
> For example:
>
> ```
> 2:CT1255:Account_Manager:1948:1684:63:1063403309:14:root.0.0.0.1:::
> ```

Interpreting Error Messages

The following examples show the typical process for evaluating and interpreting error messages to resolve problems with BRM.

Example 1: Failure of a Client Application

A BRM client application fails and displays an error message.

- Look in the application's log file. The file shows the following error message:

  ```
  E Fri Sep 12 14:50:05 2003 db2.corp:12602 sample_app.c:173
  1:CT1255:Account_Manager:1948:1684:63:1063403309:14
  op_cust_create_acct error [location=pin_errloc_dm class= errno= field num=
  recid=<0> reserved=<0>]
  ```

  The message shows that:

  - At 014:50:05 the system returned an error.
  - The host name is **db2.corp**.
  - The file name is **sample_app.c**.
  - The line of code is **173**.
  - The correlation ID is

    ```
    1:CT1255:Account_Manager:1948:1684:63:1063403309:14
    ```

  - There was a problem creating an account.
  - The error was first found in the DM.

  ```
  Check the Data Manager log file (dm_oracle.pinlog) for an error message that
  occurred at the same time and has the same correlation ID, in this case Fri Sep 12
  14:50:5 and
  1:CT1255:Account_Manager:1948:1684:63:1063403309:14:
  E Fri Sep 12 14:50:05 2003 db2.corp dm:12250 dm_subr.c(1.58):351
  1:CT1255:Account_Manager:1948:1684:63:1063403309:14
  ORACLE error: do_sql_insert: oexec: code 1653, op 4, peo 1
  =ORA-01653: unable to extend table PIN.EVENT_T by 512 in
  tablespace PIN00"insert into event_t ( poid_db, poid_id1,
  poid_id0, poid_type, aobj_db,
  ```

  The error message shows an Oracle error, with the Oracle code 1653.

- Consult the Oracle documentation. Code 1653 indicates that there is a problem with growing an extent. Because the error message reported that BRM was unable to extend

one of the tables, you can infer that the problem is that there is no more room in the database and you must increase its size, as explained in the Oracle documentation.

Example 2: Getting More Information from Error Numbers

You cannot start the DM.

- Check the **dm_oracle.pinlog** file, which shows the following error message:

```
E THU Sep 11 00:30:49 2003 kauai dm:29349 dm_main.c(1.74):1723
1:CT1255:dm:28492:1:0:1063265316:0
DM master dm_die:"bad bind(2)", errno 125
```

This error message indicates that the DM cannot initiate itself. Usually, **errno** followed by a number means that a system message is associated with this error. You can check the error file: **/usr/include/sys/errno.h**. In this case, error 125 is listed as “EADDRINUSE: Address already in use". In other words, the DM process is trying to use a port that is already in use by another process.

Example 3: Getting More Information about Oracle Errors

An error in the application log file indicates the error location is the DM.

- Check the **dm_oracle.pinlog** file, which shows the following error message:

```
E WED Aug 18 01:40:07 2003 kauai dm:402.354 dm_subr.c(1.80):481
1:CT1255:dm:28509:1:0:1061195411:7
ORACLE error: do_sql_insert: obndrv: code 1036, op 28, peo 0
=ORA-01036: illegal variable name/number
was binding ":poid_DB" buf 0x195b180, bufl 5, ftype 5
```

The message shows an Oracle error, number 1036, which you can investigate in the Oracle documentation by using the **oerr** command.

```
% oerr ora 1036
01036, 00000, "illegal variable name/number"
// *Cause: Unable to find bind context on user side
// *Action: Make sure that the variable being bound is in the sql statement.
```

The **obndrv** function is looking for the variable **:poid_DB** in the SQL statement, but the error says that it is not there.

40
Reference Guide to BRM Error Codes

Learn how to use the Oracle Communications Billing and Revenue Management (BRM) error locations, classes, and codes to help troubleshoot problems in your system.

Topics in this document:

- Interpreting BRM Error Codes
- BRM Error Locations
- BRM Error Classes
- BRM Error Codes

For information about troubleshooting BRM, including examples of error messages that use these error codes, see "Resolving Problems in Your BRM System".

Interpreting BRM Error Codes

When a BRM process has a problem, its log file displays an error message that often includes:

- Error location; for example:

  ```
  location=<PIN_ERRLOC_FM:5>
  ```

- Error class; for example:

  ```
  class=<PIN_ERRCLASS_APPLICATION:4>
  ```

- Error code; for example:

  ```
  errno=<PIN_ERR_BAD_VALUE:46>
  ```

The tables below list the error locations, classes, and codes and give the meaning of each. Use this information to help find what caused the problem and what you can do to solve it.

BRM Error Locations

Table 40-1 lists the BRM error locations.

Table 40-1 BRM Error Locations

Error Location	No.	Source of the Error
PIN_ERRLOC_PCM	**1**	General problem connecting to BRM. Common causes include illegal parameters.
PIN_ERRLOC_PCP	**2**	Internal error. The Portal Communications Protocol (PCP) library provides communication support between the modules of BRM. Common causes include network connection failures. This value indicates a system problem that requires immediate attention.

Table 40-1 (Cont.) BRM Error Locations

Error Location	No.	Source of the Error
PIN_ERRLOC_CM	**3**	Connection Manager. Common causes include an unknown opcode or an input flist missing the required Portal object ID (POID) field.
PIN_ERRLOC_DM	**4**	Data Manager. Common causes include an input flist that does not meet the required specification or a problem communicating with the underlying data storage system.
PIN_ERRLOC_FM	**5**	Facilities Module. Common causes include an input flist that does not conform to the required specification.
PIN_ERRLOC_FLIST	**6**	An flist manipulation routine local to the application. Common causes include an illegal parameter and low system memory.
PIN_ERRLOC_POID	**7**	POID manipulation routine local to the application. Common causes include an illegal parameter and low system memory.
PIN_ERRLOC_APP	**8**	An error occurred within an application.
PIN_ERRLOC_QM	**9**	An error occurred within the Queue Manager (**qmflist**).
PIN_ERRLOC_PCMCPP	**10**	An error occurred within the PCM C++ wrapper.
PIN_ERRLOC_LDAP	**11**	An error occurred within the LDAP library.
PIN_ERRLOC_UTILS	**14**	An error occurred within a BRM utility.
PIN_ERRLOC_JS	**15**	An error occurred within the Java Server Framework.
PIN_ERRLOC_JSAPP	**16**	An error occurred within the Java Server Application or opcode handler.
PIN_ERRLOC_PDO	**17**	An error occurred within the BRM data objects.

BRM Error Classes

Table 40-2 lists the BRM error classes.

Table 40-2 BRM Error Classes

Error Class	No.	Meaning
PIN_ERRCLASS_SYSTEM_DETERMINATE	**1**	The error was caused by a system failure during the operation. Retrying the operation is unlikely to succeed, and the system failure should be investigated immediately. The error was detected before any data was committed to the database; no data has changed. After the error is fixed, retry the operation.

Table 40-2 (Cont.) BRM Error Classes

Error Class	No.	Meaning
PIN_ERRCLASS_SYSTEM_INDETERMINATE	**2**	The error was caused by a system failure during the “commit" phase of an operation. The error might not be repeatable, and the system might not save specific information about the error. There is a small window during the commit where a network failure can leave the system unsure of whether the commit occurred. This means it is up to the application to determine whether system data has been changed. This class of error is extremely rare, but you must deal with it carefully to avoid corrupting the data in the database. The transactional model of BRM guarantees that either all the changes within the indeterminate transaction are committed, or none of them are committed and the system data is left unchanged. If you find that no changes were made, resolve the system failure and retry the operation.
PIN_ERRCLASS_SYSTEM_RETRYABLE	**3**	The error was probably caused by a transient condition. Try the operation again. Common causes include a temporary shortage of system resources (perhaps caused by too many connections to the CM) or a failure of a network connection that you can route around. The error was detected before any data was committed to the database; no data has changed.
PIN_ERRCLASS_APPLICATION	**4**	The error was caused by a custom application passing invalid data to BRM or by a system failure within the client application. The error was detected before the requested operation was performed, so no data in the database has been changed. After you fix the error, retry the operation.

BRM Error Codes

Table 40-3 lists the BRM error codes.

Table 40-3 BRM Error Codes

Error Code	No.	Description
PIN_ERR_NONE	**0**	No error. This error code indicates a general condition, not a specific error. Some BRM routines use this error code to indicate that the routine was successful.
PIN_ERR_NO_MEM	**1**	There was insufficient memory to complete the attempted operation. Check system memory. Also check the **shmsize** and **bigsize** values in the Data Manager configuration file (**pin.conf**). If you see this error code with a custom application, check for memory leaks. **Solaris:** Check that the shared memory for the system is at least as great as the shared memory set for BRM.
PIN_ERR_NO_MATCH	**2**	BRM couldn't find the value it was looking for. From the Data Manager, this error indicates a bad search template from the **qm_flist**. From the SDK FM, this error means that the FM cannot find the object it was told to modify.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_NOT_FOUND	**3**	BRM couldn't find a value. This error code doesn't always indicate an error. For example, some opcodes look for a value in the configuration file, but if the value is not there, a default value can be used.
PIN_ERR_BAD_ARG	**4**	A required field in an flist is incorrect. This is always a serious error because the system will not work until the argument is fixed. The problem is usually a programming or data entry error.
PIN_ERR_BAD_XDR	**5**	The application unexpectedly lost the connection to the BRM database. Usually, this error means that the connection to the network was lost. If the network is working correctly, the BRM server might have stopped working. Look for errors in the CM log file. Also check for CM or DM **core** files.
PIN_ERR_BAD_FIRST_READ	**6**	No longer used.
PIN_ERR_BAD_READ	**7**	BRM couldn't read from the network or some other IO device, probably because the connection was cut off unexpectedly.
PIN_ERR_NO_SOCKET	**8**	BRM couldn't create a socket. The machine or process might be overloaded and have reached a limit on socket/file descriptors. The networking part of the operating system might have failed. Try restarting the machine. If that doesn't work, report the problem to the OS vendor.
PIN_ERR_BAD_TYPE	**9**	BRM encountered an erroneous field or object type. This error usually results from programming errors in custom applications and FMs, but the error might also result from a mismatch between a field and its corresponding field type. If seen when **pcpxdr_fld_list** is called, it means a down-level version of the libraries saw incompatible wire protocols or a new field type.
PIN_ERR_DUPLICATE	**10**	BRM could not create an object because the requested ID is already used by another object.
PIN_COMPARE_EQUAL	**11**	This code doesn't indicate an error. The billing FM uses this code for internal operations.
PIN_COMPARE_NOT_EQUAL	**12**	This code doesn't indicate an error. The billing FM uses this code for internal operations.
PIN_ERR_MISSING_ARG	**13**	A required argument is missing. If the log file doesn't indicate the field, see the specification for the opcode.
PIN_ERR_BAD_POID_TYPE	**14**	BRM encountered an erroneous object type. This is similar to error 9, but is more specific to object type. For example, BRM was expecting an account object but encountered an event object.
PIN_ERR_BAD_CRYPT	**15**	Packet header failed encryption/decryption. Possible corruption of data.
PIN_ERR_BAD_WRITE	**16**	Error while attempting to send data on the IP socket.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_DUP_SUBSTRUCT	**17**	Information not available.
PIN_ERR_BAD_SEARCH_ARG	**18**	A required field in a search template or flist is incorrect. This is always a serious error because the system will not work until the argument is fixed. The problem is usually a programming or data entry error.
PIN_ERR_BAD_SEARCH_ARG_RECID	**19**	Invalid automatic number identification (ANI) in the search operation.
PIN_ERR_DUP_SEARCH_ARG	**20**	There are duplicate entries in the search argument list.
PIN_ERR_NONEXISTANT_POID	**21**	BRM cannot find the object in the database.
PIN_ERR_POID_DB_IS_ZERO	**22**	The database number is specified as zero. Make sure the routine is passing a valid database number that matches the number in the configuration file.
PIN_ERR_UNKNOWN_POID	**23**	The POID does not contain valid information, or the format is incorrect.
PIN_ERR_NO_SOCKETS	**24**	BRM couldn't create a socket. The machine or process might be overloaded and have reached a limit on socket/file descriptors. The networking part of the operating system might have failed. Try restarting the machine. If that doesn't work, report the problem to the OS vendor.
PIN_ERR_DM_ADDRESS_LOOKUP _FAILED	**25**	The Connection Manager couldn't find the Data Manager. The Bind (or DNS) service is pointing to the wrong TCP/IP address, or the network is having problems. Try pinging the DM to verify the network connection. Check the DM pointer specified in the configuration file for the CM.
PIN_ERR_DM_CONNECT_FAILED	**26**	BRM couldn't connect to the Data Manager. The configuration file for the CM might be pointing to the wrong DM, or the DM might not be running.
PIN_ERR_NAP_CONNECT_FAILED	**27**	An application couldn't connect to the Connection Manager. The configuration file for the application might be pointing to the wrong CM. The CM might not be running, or there might not be any more available connections. Also check for network errors between the application and the CM. For example, no more TCP sockets might be available on the client or CM machine.
PIN_ERR_INVALID_RECORD_ID	**28**	The ID of the specified element in the array is not valid. The specified ID might be greater than the maximum record ID.
PIN_ERR_STALE_CONF	**29**	BRM found outdated values in **pin.conf** entries.
PIN_ERR_INVALID_CONF	**30**	Configuration data is missing or in an invalid format in the **pin.conf** files.
PIN_ERR_WRONG_DATABASE	**31**	The database number in the POID is not valid.
PIN_ERR_DUP_ARG	**32**	The flist has duplicate fields or elements.
PIN_ERR_BAD_SET	**33**	BRM couldn't assign an object ID.
PIN_ERR_BAD_CREATE	**34**	A routine couldn't create an object.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_BAD_FIELD_NAME	**35**	Mapping error from field home to type.
PIN_ERR_BAD_OPCODE	**36**	Undefined opcode used.
PIN_ERR_TRANS_ALREADY_OPEN	**37**	An application attempted to open a transaction when one was already open on the same object. This error usually results from a programming error in a custom application or FM. The error sometimes appears in a series of cascading errors; look for a predecessor error.
PIN_ERR_TRANS_NOT_OPEN	**38**	An application attempted to commit or cancel a transaction, but none had been opened. This error usually results from a programming error in a custom application or FM. The error sometimes shows up in a series of cascading errors; look for a predecessor error.
PIN_ERR_NULL_PTR	**39**	A routine couldn't get a value because it was set to “null". This error usually results from a programming error in a custom application or FM.
PIN_ERR_BAD_FREE	**40**	A routine tried, but failed, to free memory that was no longer needed. This error usually results from a programming error in a custom application or FM. This problem might cause memory leaks.
PIN_ERR_FILE_IO	**41**	Error while attempting to do an IO operation on a file.
PIN_ERR_NONEXISTANT_ELEMENT	**42**	The array in the specified object doesn't have the specified element.
PIN_ERR_STORAGE	**43**	The database returned an error. Check the Data Manager log file for database-specific error messages. Also check the database server error logs.
PIN_ERR_TRANS_TOO_MANY_POIDS	**44**	BRM attempted transactions to too many Data Managers.
PIN_ERR_TRANS_LOST	**45**	The transaction was lost. The Data Manager failed during a transaction.
PIN_ERR_BAD_VALUE	**46**	BRM couldn't interpret data from the database. The data isn't valid in the current context, and BRM cannot resolve the conflict.
PIN_ERR_PARTIAL	**47**	When sending a batch of transactions to the credit card Data Managers, some of the credit cards were processed, but others were not.
PIN_ERR_NOT_YET_DONE	**48**	BRM has not yet completed an operation (such as an opcode or transaction). This is typically an internal debugging error code that isn't displayed.
PIN_ERR_STREAM_IO	**49**	The application encountered an error while sending data to or from the BRM database. Usually, this error means that the connection to the network was lost. If the network is working correctly, the server might have stopped working. Look for errors in the CM log file. Also check for CM or DM core files.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_STREAM_EOF	**50**	The application unexpectedly lost the connection to the BRM database. Usually, this error means that the connection to the network was lost. If the network is working correctly, the server might have stopped working. Look for errors in the CM log file. Also check for CM or DM **core** files.
PIN_ERR_OP_NOT_OUTSTANDING	**51**	No operation is in progress under this context.
PIN_ERR_OP_ALREADY_BUSY	**52**	There is an operation already in progress within this context.
PIN_ERR_OP_ALREADY_DONE	**53**	Certain operations can be done only once. This error indicates that an operation of this type is being attempted a second time.
PIN_ERR_NO_DATA_FIELDS	**54**	The Data Manager received an flist with no data fields. Check the input flist.
PIN_ERR_PROHIBITED_ARG	**55**	BRM couldn't create an object because one or more values were invalid or fields do not allow updating.
PIN_ERR_BAD_LOGIN_RESULT	**56**	The application couldn't connect to the CM. Check the login name and password.
PIN_ERR_CM_ADDRESS_LOOKUP _FAILED	**57**	The application couldn't find the computer running the CM. The Bind (or DNS) service is pointing to the wrong TCP/IP address, or the network is having problems. Try pinging the CM to verify the network connection. Check the CM host name specified in the configuration file for the application.
PIN_ERR_BAD_LOGIN_REDIRECT_INFO	**58**	The re direction information received from the CMMP is incorrect.
PIN_ERR_TOO_MANY_LOGIN _REDIRECTS	**59**	Too many connect login redirects from the CMMP. This error might have resulted from a loop in the configuration. Check the configuration files for the application, CM, and DM.
PIN_ERR_STEP_SEARCH	**60**	The step search operation didn't find an expected **STEP_NEXT/ STEP_END**.
PIN_ERR_STORAGE_DISCONNECT	**61**	BRM lost the connection with the database during the middle of a transaction and couldn't re-establish the connection. See "Configuring Oracle Databases" in *BRM Installation Guide*.
PIN_ERR_NOT_GROUP_ROOT	**62**	Not the root of a group.
PIN_ERR_BAD_LOCKING	**63**	Error while attempting to lock/unlock a heap storage.
PIN_ERR_AUTHORIZATION_FAIL	**64**	No information available.
PIN_ERR_NOT_WRITABLE	**65**	Tried to write a field that cannot be modified.
PIN_ERR_UNKNOWN_EXCEPTION	**66**	Unknown C++ exception.
PIN_ERR_START_FAILED	**67**	BRM couldn't start the process.
PIN_ERR_STOP_FAILED	**68**	BRM couldn't stop the process.
PIN_ERR_INVALID_QUEUE	**69**	No information available.
PIN_ERR_TOO_BIG	**70**	No information available.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_BAD_LOCALE	**71**	BRM doesn't understand the locale. Check the locale of the computer running the client application.
PIN_ERR_CONV_MULTIBYTE	**72**	A client application had a problem converting data from UTF8 format, as it is stored in the BRM database, to multibyte format. Either the client has the wrong locale or the data is corrupted.
PIN_ERR_CONV_UNICODE	**73**	A client application had a problem converting data from UTF8 format, as it is stored in the BRM database, into Unicode format. Either the client has the wrong locale or the data are corrupted.
PIN_ERR_BAD_MBCS	**74**	The input flist includes a string that is not in valid multibyte format.
PIN_ERR_BAD_UTF8	**75**	The input flist includes a string that is not in valid Unicode format.
PIN_ERR_CANON_CONV	**76**	No information available.
PIN_ERR_UNSUPPORTED_LOCALE	**77**	BRM doesn't support canonicalization for the locale of the client application.
PIN_ERR_CURRENCY_MISMATCH	**78**	A nonpaying bill unit (**/billinfo** object) or account has a different account currency than the parent account.
PIN_ERR_DEADLOCK	**79**	Two or more database sessions attempted to access the same database resource. Each session waits for another session to release locks on the resource. The database detects the deadlock and cancels one of the session's operations. If you receive this error, retry the transaction or operation.
PIN_ERR_BACKDATE_NOT_ALLOWED	**80**	BRM cannot backdate the adjustment, write-off, or other transaction because the G/L report has already been posted.
PIN_ERR_CREDIT_LIMIT_EXCEEDED	**81**	No information available.
PIN_ERR_IS_NULL	**82**	The value is **Null** (not set).
PIN_ERR_DETAILED_ERR	**83**	A detailed error message uses an enhanced buffer (**errbuf**).
PIN_ERR_PERMISSION_DENIED	**84**	The attempted operation is not allowed; data is not viewable.
PIN_ERR_PDO_INTERNAL	**85**	An internal error occurred in the BRM data objects.
PIN_ERR_IPT_DNIS	**86**	The Dialed Number Identification Service (DNIS) is not authorized.
PIN_ERR_DB_MISMATCH	**87**	The database numbers do not match.
PIN_ERR_NO_CREDIT_BALANCE	**88**	No credit balance is available.
PIN_ERR_NOTHING_TO_BILL	**89**	There are no new items to bill.
PIN_ERR_MASTER_DOWN	**90**	The main BRM system is down.
PIN_ERR_OPCODE_HNDLR_INIT_FAI	**91**	The opcode handler initialization at JS failed.
PIN_ERR_STMT_CACHE	**92**	There is a problem with the statement cache.
PIN_ERR_CACHE_SIZE_ZERO	**93**	Tried to init **cm_cache** with zero size.
PIN_ERR_INVALID_OBJECT	**94**	The POID is invalid. This error occurs when an object is accessed whose *database_number* field of the POID is NULL.
PIN_ERR_VALIDATE_ADJUSTMENT	**95**	The validate adjustment policy opcode fails.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_SYSTEM_ERROR	**96**	A generic system error during running application.
PIN_ERR_BILLING_ERROR	**97**	An error occurred during the billing run.
PIN_ERR_AUDIT_COMMIT_FAILED	**98**	Commit for the audit tables failed.
PIN_ERR_NOT_SUPPORTED	**99**	The operation is not supported.
PIN_ERR_INVALID_SER_FORMAT	**100**	The serialized format received from the database is incorrect.
PIN_ERR_READ_ONLY_TXN	**101**	Cannot insert or update in a read-only transaction.
PIN_ERR_VALIDATION_FAILED	**102**	Bundle or package validation failed.
PIN_ERR_PROC_BIND	**103**	The binding for the Procedure arguments failed.
PIN_ERR_PROC_EXEC	**104**	The procedure returned an Application Specific error.
PIN_ERR_STORAGE_FAILOVER	**105**	A RAC failover message.
PIN_ERR_RETRYABLE	**106**	A failover error occurred and is retryable if needed.
PIN_ERR_NOT_PRIMARY	**107**	Not the primary instance.
PIN_ERR_TIMEOUT	**108**	Request timeout.
PIN_ERR_CONNECTION_LOST	**109**	The connection has been lost.
PIN_ERR_BAD_ENC_SCHEME	**116**	The MD5 encryption scheme is configured; however, the AES encryption library is being used.
PIN_ERR_MAX_BILL_WHEN	**117**	The value in the **bill_when** field exceeds the maximum number of months allowed in billing cycles.
PIN_ERR_PERF_LIMIT_REACHED	**118**	The performance limit has been reached.
PIN_ERR_ACTIVE_NOT_READY	**119**	The TIMOS passive instance is switching over to active, but it is not active yet.
PIN_ERR_POID_ALREADY_LOCKED	**120**	The POID is already locked.
PIN_ERR_SERVICE_LOCKED	**121**	This error code is used by SOX.
PIN_ERR_XAER_RMFAIL	**122**	The extended architecture (XA) transaction resource manager (JCA Resource Adapter) is unavailable.
PIN_ERR_XAER_RMERR	**123**	An XA transaction resource manager (JCA Resource Adapter) error occurred in the transaction branch.
PIN_ERR_XAER_PROTO	**124**	An XA transaction protocol error has occurred; a routine was started in an improper context.
PIN_ERR_XAER_OUTSIDE	**125**	The resource manager (JCA Resource Adapter) is doing work outside the XA transaction.
PIN_ERR_XAER_NOTA	**126**	The global transaction ID (XID) for the XA transaction is invalid.
PIN_ERR_XAER_INVAL	**127**	An invalid argument was passed during the XA transaction.
PIN_ERR_XAER_DUPID	**128**	The global transaction ID (XID) for the XA transaction already exists.
PIN_ERR_XAER_ASYNC	**129**	An asynchronous operation is outstanding.
PIN_ERR_XA_RDONLY	**130**	The XA transaction branch was read-only and has been committed.
PIN_ERR_XA_RETRY	**131**	The routine returned without having any effect. It can be reissued.

Table 40-3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_XA_HEURHAZ	**132**	The XA transaction branch might have been manually committed to the BRM database. Some parts of the transaction are known to have been manually committed or rolled back, but the outcome of the entire transaction is unknown.
PIN_ERR_XA_HEURCOM	**133**	The XA transaction branch has been manually committed to the BRM database. BRM returns this error code to JCA Resource Adapter during the recovery process of a failed two-phase commit transaction.
PIN_ERR_XA_HEURRB	**134**	The XA transaction branch has been manually rolled back. BRM returns this error code to JCA Resource Adapter during the recovery process of a failed two-phase commit transaction.
PIN_ERR_XA_HEURMIX	**135**	Part of the XA transaction branch has been manually committed to the BRM database, and part has been manually rolled back. BRM returns this error code to JCA Resource Adapter during the recovery process of a failed two-phase commit transaction.
PIN_ERR_XA_RBCOMMFAIL	**136**	The XA transaction was rolled back because of a communications failure.
PIN_ERR_XA_RBDEADLOCK	**137**	The XA transaction was rolled back because a deadlock was detected.
PIN_ERR_XA_RBINTEGRITY	**138**	The XA transaction was rolled back because a condition that violates the integrity of the resource was detected.
PIN_ERR_XA_RBOTHER	**139**	The XA transaction was rolled back for a reason not specified by an error code in this table.
PIN_ERR_XA_RBPROTO	**140**	The XA transaction was rolled back because of a protocol error in the resource manager.
PIN_ERR_XA_RBROLLBACK	**141**	The XA transaction was rolled back for an unspecified reason.
PIN_ERR_XA_RBTIMEOUT	**142**	The XA transaction was rolled back because it timed out.
PIN_ERR_XA_RBTRANSIENT	**143**	The XA transaction was rolled back because of a brief malfunction. The transaction branch can be retried.

Part V

Monitoring BRM with Prometheus and Grafana

This part describes how to monitor an Oracle Communications Billing and Revenue Management (BRM) system using external applications, such as Prometheus and Grafana. It contains the following chapters:

- Monitoring BRM Components
- Monitoring Business Operations Center
- Monitoring Billing Care and Billing Care REST API

41

Monitoring BRM Components

Learn how to use external applications, such as Pushgateway, Prometheus, and Grafana, to monitor components in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Monitoring Your BRM Components

You set up the monitoring of your BRM components by using the following external applications:

- **Pushgateway**: Pushgateway caches your BRM metrics and exposes them to Prometheus.
- **Prometheus**: Prometheus scrapes your BRM metrics from Pushgateway and then stores them in a time-series database.
- **Grafana**: Use this open-source tool to view on a graphical dashboard all BRM metric data stored in Prometheus.

Setting Up Monitoring for BRM Components

To set up monitoring for your BRM components:

1. Install the following external software on your system:
   - Prometheus Pushgateway. See https://github.com/prometheus/pushgateway on the GitHub website.
   - Prometheus. See "Installation" in the Prometheus documentation.
   - Grafana. See "Install Grafana" in the Grafana documentation.

   For the list of compatible software versions, see "BRM Software Compatibility" in *BRM Compatibility Matrix*.
2. Optionally, customize Pushgateway. See "Customizing Pushgateway for BRM".
3. Enable Perflib-based monitoring of your CM. See "Enabling Monitoring of Your CM".
4. Enable Perflib-based monitoring of your Oracle DM. See "Enabling Monitoring of Your Oracle DM".
5. Enable Perflib-based monitoring of the Account Synchronization DM and Synchronization Queue DM. See "Enabling Monitoring of dm_ifw_sync and dm_aq".
6. Expose metrics for BRM Java applications such as Batch Controller and REL Daemon. See "Enabling Monitoring of BRM Java Applications".

7. Expose metrics for Web Services Manager through WebLogic Monitoring Exporter. See "Enabling Monitoring of Web Services Manager".
8. Configure Prometheus to scrape BRM metric data from the Pushgateway endpoint. See "Configuring Prometheus for BRM Components".
9. Configure Grafana to display metric data for your BRM components. See "Creating Grafana Dashboards for BRM Components".

Customizing Pushgateway for BRM

By default, Pushgateway exposes BRM metrics at **http://localhost:9091/metrics**, and Prometheus scrapes the **brm_status_tracker** and **brm_memory_usage** metrics from Pushgateway every 5 seconds. However, you can customize where Pushgateway exposes the BRM metrics and how Prometheus scrapes them.

To customize Pushgateway for BRM:

1. Open the *BRM_home***/bin/configurations.yaml** file in a text editor.
2. To customize where Pushgateway exposes BRM metrics, edit these keys:
   - **pushgateway.host**: The hostname of the machine on which to deploy Pushgateway. The default is **localhost**.
   - **pushgateway.port**: The port number for Pushgateway. The default is **9091**.
   - **pushgateway.protocol**: The protocol type, such as **http** or **https**. The default is **http**.
3. To customize the metric names and scrape intervals for Prometheus, modify these keys:
   - **prometheus.service_monitoring.brm_status_metric_name**: The name of the metric for monitoring the current status of BRM services.
   - **prometheus.service_monitoring.scrape_interval**: How frequently Prometheus scrapes the BRM service status metric, in seconds.
   - **prometheus.memory_monitoring.brm_status_metric_name**: The name of metric for monitoring the CPU memory used by BRM services.
   - **prometheus.memory_monitoring.scrape_interval**: How frequently Prometheus scrapes the CPU memory status metric, in seconds.
4. Save and close the file.

Example 41-1 Customizing the Pushgateway Protocol and Prometheus Scrape Interval

The **configurations.yaml** entries in this example would change the Pushgateway protocol to https and the Prometheus scrape interval for both metrics to 1 second.

```
pushgateway:
   host: localhost
   port: 9091
   protocol: "https"

prometheus:
   service_monitoring:
      brm_status_metric_name: "brm_status_tracker"
      scrape_interval: 1
```

```
memory_monitoring:
   brm_status_metric_name: "brm_memory_usage"
   scrape_interval: 1
```

Enabling Monitoring of Your CM

To enable monitoring of your CM:

1. In your CM configuration file (*BRM_home*/**sys/cm/pin.conf**), edit the following Perflib-related entries:
   - In the **fm_perflib_config** entry, enter the full path to your **fm_perflib.so** file:

     - cm fm_module *Path*/**fm_perflib.so fm_perflib_config - pin**

   - In the **perflib_monitor_file** entry, enter the full path to your Perflib data file:

     - perflib_monitor_file perflib_data *Path*/**perflib_data.dat**

   - In the **perflib_pin_shlib** entry, enter the full path to the shared library in which core BRM functionality for the CMs and applications are defined:

     - perflib perflib_pin_shlib *Path*/**libcmpin.so**

2. In the *PERFLIB_home*/**perf.env** file, set the Perflib timing-related entries in Table 41-1.

   Table 41-1 Perflib Timing Entries

Entry Name	Description
PERFLIB_VAR_TIME	Whether Perflib tracing is activated immediately: • **0**: Timing is disabled at startup. • **1**: Real-time timing is enabled at startup. This is the default. • **2**: File-based timing is enabled at startup. • **3**: File-based and real-time timing is enabled at startup.
PERFLIB_VAR_FLIST	Whether Perflib flist tracing is activated immediately: • **0**: Flist logging is disabled. This is the default. • **1**: Summary logging is enabled at startup. • **2**: Full flist logging is enabled at startup.
PERFLIB_VAR_ALARM	Whether Perflib alarm functionality is activated immediately: • **0**: Alarms are disabled at startup. • **1**: Alarms are enabled at startup. This is the default.

3. Start the CM process by running the following command. Ignore the LD_PRELOAD errors.

   ./start_it start_cm

4. Use **testnap** to run a few opcodes. For example:

```
xop PCM_OP_SEARCH 0x10 1
```

   For more information about **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*.

5. Run **pstatus** to verify that opcode metrics are getting captured in the Perflib data file:

```
pstatus -s10 perflib_data.dat
```

   If successful, you should see something similar to this:

```
Timestamp            Program     Opcode  Opcode Name
Calls    Errors    Records     Elapsed       OpAvg      SysAvg
SysRate/s     PID
---------            -------     ------  -----------
-----    ------    -------     -------       -----      ------
---------     ---
14/07/2021 04:07:34  testnap        155  PCM_OP_ACT_LOGIN
2         0          0    0.288688    0.144344         inf
0.00    9250
14/07/2021 04:07:34  testnap          3  PCM_OP_READ_OBJ
2         0          2    0.052501    0.026251         inf
0.00    9250
14/07/2021 04:07:34  testnap        156  PCM_OP_ACT_LOGOUT
2         0          0    0.043381    0.021691         inf
0.00    9250
```

6. To expose Prometheus metrics for the CM, run the **start_monitor** script:

```
./start_monitor -P 9090 perflib_data.dat,cm
```

7. Run the following cURL command to view the exposed metrics:

```
curl http://localhost:9090/metrics
```

   If successful, you should see something similar to this:

```
brm_opcode_calls_total{application="cm", opcode="20", opflags="0",
program_name="cm", object_type="", opcode_name="PCM_OP_GET_DD"} 6
brm_opcode_errors_total{application="cm", opcode="20", opflags="0",
program_name="cm", object_type="", opcode_name="PCM_OP_GET_DD"} 0
brm_opcode_records_total{application="cm", opcode="20",
opflags="0", program_name="cm", object_type="",
opcode_name="PCM_OP_GET_DD"} 0
brm_opcode_exec_time_total{application="cm", opcode="20",
opflags="0", program_name="cm", object_type="",
opcode_name="PCM_OP_GET_DD"} 1.371886758
brm_opcode_user_cpu_time_total{application="cm", opcode="20",
opflags="0", program_name="cm", object_type="",
opcode_name="PCM_OP_GET_DD"} 0.019318000
```

```
brm_opcode_system_cpu_time_total{application="cm", opcode="20",
opflags="0", program_name="cm", object_type="",
opcode_name="PCM_OP_GET_DD"} 0.006629000
```

Enabling Monitoring of Your Oracle DM

To enable monitoring of your Oracle DM:

1. In your Oracle DM configuration file (*BRM_home*/**sys/cm/pin.conf**), set the **perflib_enabled** entry to **1**:

   ```
   - dm perflib_enabled 1
   ```

2. In the *PERFLIB_home*/**perf_dm_oracle.env** file, set the Perflib timing-related entries in Table 41-1.
3. Start the dm_oracle process by running the following command. Ignore the LD_PRELOAD errors.

   ```
   ./start_it start_dm_oracle
   ```

4. Start **testnap** and then run a few opcodes. For example:

   ```
   xop PCM_OP_READ_OBJ 0x0040 1
   ```

   For more information about **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*.
5. Run **pstatus** to verify that opcode metrics are getting captured in the Perflib data file:

   ```
   pstatus -s10 perflib_dm_oracle.dat
   ```

After you start the monitoring process, the Oracle DM metrics will be exposed at the following endpoint:

```
http://localhost:9091/metrics
```

Enabling Monitoring of dm_ifw_sync and dm_aq

To enable monitoring of your Account Synchronization DM (**dm_ifw_sync**) and Synchronization Queue DM (**dm_aq**):

1. In your **dm_ifw_sync** configuration file (*BRM_home*/**sys/dm_ifw_sync/pin.conf**), set the **perflib_enabled** entry to **1**:

   ```
   - dm perflib_enabled 1
   ```

2. In your **dm_aq** configuration file (*BRM_home*/**sys/dm_aq/pin.conf**), set the **perflib_enabled** entry to **1**:

   ```
   - dm perflib_enabled 1
   ```

Alternatively, you can enable monitoring of **dm_oracle**, **dm_ifw_sync**, and **dm_aq** by setting the following entry to **1** in your *Perflib_home***/perf_dm.env** file:

```
DM_PERFLIB_ENABLED=1
```

To temporarily disable the monitoring of **dm_oracle**, **dm_ifw_sync**, or **dm_aq**, set its environment variable to **0** in its respective start script. For example, to temporarily disable the monitoring of **dm_ifw_sync**, you would modify the following line in bold in the **start_dm_ifw_sync** file:

```
if [ -f ${DMPERFLIBENV} ]; then
source ${DMPERFLIBENV}
export PERFLIB_DATA_FILE=$PERFLIB_HOME/perflib_dm_ifw_sync_data.dat
export DM_PERFLIB_ENABLED=0
fi
```

After you start the monitoring process, the dm_ifw_sync and dm_aq metrics will be exposed at the following endpoint:

```
http://localhost:3000/metrics
```

Enabling Monitoring of BRM Java Applications

You use the JMX framework to expose metrics for BRM Java applications, such as Batch Controller, REL daemon, and EAI Java Server (JS), in Prometheus format.

To enable the monitoring of BRM Java applications:

1. Download the latest JMX Exporter **.jar** file from https://github.com/prometheus/jmx_exporter.

   For a list of compatible versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.

2. Create a configuration file named **config.yaml** in your *BRM_home***/bin** directory.

3. Configure the JMX Exporter by adding the parameters defined in https://github.com/prometheus/jmx_exporter#configuration.

4. To configure BRM Java applications to expose JMX metrics in Prometheus format, edit the component's start script to include the JMX_EXPORTER_OPTS environment variable.

   For example, you would expose JMX metrics for EAI JS by adding the following lines to the **start_eai_js** script:

   ```
   JMX_EXPORTER_OPTS="-javaagent:/scratch/jmx_prometheus_javaagent-
   version.jar=12347:/scratch/config.yaml"
   $JAVA -Deai_js=1 ${JMX_EXPORTER_OPTS} -mx256m -ms64m -ss1m
   com.portal.js.JSMain >>${JSLOG} 2>&1 &
   ```

   where *version* is the JMX Exporter version number.

   The JMX metrics in Prometheus format will be available at the **/metrics** endpoint using the port configured in the component's start script. In this example, the endpoint for EAI JS metrics would be **http://localhost:12347/metrics**.

Enabling Monitoring of Web Services Manager

You use the WebLogic Monitoring Exporter to expose metrics for Web Services Manager in Prometheus format.

To enable monitoring of Web Services Manager:

1. Download the latest supported version of WebLogic Monitoring Exporter (**get*N.N*.sh**) from https://github.com/oracle/weblogic-monitoring-exporter/releases, where *N.N* is the version number.

   For a list of compatible versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.

2. Edit the **wls-exporter-config.yaml** file to include the metrics to scrape from BRM. For the list of supported metrics, see "WebLogic-Based Application Metrics".

   For example:

```
metricsNameSnakeCase: true
domainQualifier: true
#restPort: 7001
queries:
- key: name
  keyName: server
  applicationRuntimes:
    key: name
    keyName: app
    componentRuntimes:
      type: WebAppComponentRuntime
      prefix: webapp_config_
      key: name
      values: [deploymentState, contextRoot, sourceInfo,
openSessionsHighCount, openSessionsCurrentCount,
sessionsOpenedTotalCount, sessionCookieMaxAgeSecs,
sessionInvalidationIntervalSecs, sessionTimeoutSecs,
singleThreadedServletPoolSize, sessionIDLength, servletReloadCheckSecs,
jSPPageCheckSecs]
      servlets:
        prefix: wls_servlet_
        key: servletName
        values: invocationTotalCount

- JVMRuntime:
    prefix: wls_jvm_
    key: name

- executeQueueRuntimes:
    prefix: wls_socketmuxer_
    key: name
    values: [pendingRequestCurrentCount]

- workManagerRuntimes:
    prefix: wls_workmanager_
    key: name
    values: [stuckThreadCount, pendingRequests, completedRequests]
```

```
- threadPoolRuntime:
    prefix: wls_threadpool_
    key: name
    values: [executeThreadTotalCount, queueLength,
stuckThreadCount, hoggingThreadCount]
```

3. Run the following command:

   bash get*N.N.***sh wls-exporter-config.yaml**

   The **wls-exporter.war** is downloaded to your current directory.

4. Deploy the **wls-exporter.war** in the same WebLogic domain as **BrmWebServices.war** or **infranetwebsvc.war**.

WebLogic Monitoring Exporter exposes Web Services Manager metrics in Prometheus format to the **http://localhost:8080/wls-exporter/metrics** endpoint on the WebLogic Server.

Configuring Prometheus for BRM Components

To configure Prometheus to scrape metric data from your Pushgateway endpoint:

1. Edit your **prometheus.yaml** file to include:
   - The targets to scrape for your components that use Perflib-based monitoring (CM, Oracle DM, dm_aq, and dm_ifw_sync), REL daemon, Batch Controller, and BRM REST Services Manager, and optionally, node exporter.
   - Prometheus Alertmanager configuration.

   For example:

```
global:
  scrape_interval: 1s
  evaluation_interval: 15s

alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - alertmanager_host:9093

rule_files:
  - "first_rules.yaml"
  - "second_rules.yaml"

scrape_configs:
  - job_name: 'prometheus'

    static_configs:
    - targets: ['localhost:9090','localhost:9091','localhost:3000']

  - job_name: 'perflib'

    static_configs:
    - targets: ['localhost:12345']

  - job_name: 'rel_daemon'
```

```
    static_configs:
    - targets: ['localhost:12346']

  - job_name: 'batch_controller'

    static_configs:
    - targets: ['localhost:12347']

  - job_name: 'brm-rest-services-manager'

    static_configs:
    - targets: ['localhost:12348']

  - job_name: 'node-exporter'

    static_configs:
    - targets: ['localhost:12349']
```

For information about editing this file, see "Prometheus Configuration" in the Prometheus documentation.

2. Configure the alert rules in Prometheus.
To do so, add alert rules for your components that are similar to the ones shown below to the rules files referenced in **prometheus.yaml**.

```
groups:
  - name: brm-rsm-alert-rules
     rules:
      - alert: CPU_UsageWarning
        annotations:
          message: CPU has reached 80% utilization
        expr: avg without(cpu) (rate(node_cpu_seconds_total{job="node-
exporter", instance="node_exporter_host:node_exporter_port", mode!="idle"}
[5m])) > 0.8
        for: 5m
        labels:
          severity: critical
      - alert: Memory_UsageWarning
        annotations:
          message: Memory has reached 80% utilization
        expr: node_memory_MemTotal_bytes{job="node-exporter",
instance="node_exporter_host:node_exporter_port"}
              - node_memory_MemFree_bytes{job="node-exporter",
instance="node_exporter_host:node_exporter_port"}
              - node_memory_Cached_bytes{job="node-
exporter",instance="node_exporter_host:node_exporter_port"}
              - node_memory_Buffers_bytes{job="node-exporter",
instance="1"} > 22322927872
        for: 5m
        labels:
          severity: critical
```

For more information about defining alert rules, see "Alerting Rules" in the Prometheus documentation.

> **Note:**
>
> You can also configure alert rules and add or remove email recipients in the Grafana user interface. See "Legacy alerting" in the Grafana documentation for more information.

3. Restart Prometheus by running this command:

   ```
   ./prometheus --config.file=prometheus.yaml
   ```

4. Start monitoring BRM services for their current status by running this command:

   ```
   start_service_monitor
   ```

5. Start monitoring the CPU and memory used by BRM services by running this command:

   ```
   start_memory_monitor
   ```

To ensure that you configured Prometheus correctly and that it has started scraping BRM metric data, go to the following URL: **http://localhost:9090/graph**.

> **Note:**
>
> If you need to stop the service and CPU monitors for any reason, run the following commands:
>
> ```
> stop_service_monitor
> stop_memory_monitor
> ```

Creating Grafana Dashboards for BRM Components

You can create a dashboard in Grafana for displaying the metric data for your BRM components.

Alternatively, you can use the sample dashboards that are included with the BRM SDK package. To use the sample dashboards, import the JSON files from the *BRM_home*/**PortalDevKit/source/samples/dashboards** directory into Grafana. Table 41-2 describes each sample dashboard.

Table 41-2 Sample Grafana Dashboards

File Name	Description
ocbrm-batch-controller-dashboard.json	Allows you to view JVM-related metrics for the Batch Controller.
ocbrm-cm-dashboard.json	Allows you to view CPU and opcode-level metrics for the CM.
ocbrm-dm-oracle-dashboard.json	Allows you to view CPU and opcode-level metrics for the Oracle DM.

Table 41-2 (Cont.) Sample Grafana Dashboards

File Name	Description
ocbrm-dm-oracle-shm-dashboard.json	Allows you to view shared memory, front-end, and back-end metrics for the Oracle DM.
ocbrm-eai-js-dashboard.json	Allows you to view JVM and opcode-related metrics for the EAI JS.
ocbrm-services-dashboard.json	Allows you to view metrics regarding the status and memory usage of BRM services.
ocrsm-rsm-dashboard.json	Allows you to view standard Helidon MP monitoring metrics for BRM REST Services Manager.

For information about importing dashboards into Grafana, see "Export and Import" in the *Grafana Dashboards* documentation.

BRM Opcode Metrics

Table 41-3 describes the metrics for retrieving runtime information for BRM opcodes.

Table 41-3 BRM Opcode Metrics

Metric Name	Metric Type	Description	BRM Service
brm_opcode_calls_total	Counter	The total number of calls to a BRM opcode.	cm dm_oracle dm_ifw_sync dm_aq
brm_opcode_errors_total	Counter	The total number of errors when executing a BRM opcode.	cm dm_oracle dm_ifw_sync dm_aq
brm_opcode_exec_time_total	Counter	The total time taken to execute a BRM opcode.	cm dm_oracle dm_ifw_sync dm_aq
brm_opcode_user_cpu_time_total	Counter	The total CPU time taken to execute the BRM opcode in user space.	cm dm_oracle dm_ifw_sync dm_aq
brm_opcode_system_cpu_time_total	Counter	The total CPU time taken to execute the BRM opcode in OS kernel space.	cm dm_oracle dm_ifw_sync dm_aq

Table 41-3 (Cont.) BRM Opcode Metrics

Metric Name	Metric Type	Description	BRM Service
brm_opcode_records_total	Counter	The total number of records returned by the BRM opcode execution.	cm dm_oracle dm_ifw_sync dm_aq
brm_dmo_shared_memory_used_current	Gauge	The total number of shared memory blocks currently used by dm_oracle.	cm
brm_dmo_shared_memory_used_max	Counter	The maximum number of shared memory blocks currently used by dm_oracle.	cm
brm_dmo_shared_memory_free_current	Gauge	The total number of free shared memory blocks available to dm_oracle.	cm
brm_dmo_shared_memory_hwm	Gauge	The shared memory high-water mark for dm_oracle.	cm
brm_dmo_shared_memory_bigsize_used_max	Counter	The maximum big size shared memory used by dm_oracle in bytes.	cm
brm_dmo_shared_memory_bigsize_used_current	Gauge	The total big size shared memory used by dm_oracle in bytes.	cm
brm_dmo_shared_memory_bigsize_hwm	Gauge	The big size shared memory high-water mark for dm_oracle in bytes.	cm
brm_dmo_front_end_connections_total	Gauge	The total number of connections for a dm_oracle front-end process.	cm
brm_dmo_front_end_max_connections_total	Counter	The maximum number of connections for a dm_oracle front-end process.	cm
brm_dmo_front_end_trans_done_total	Counter	The total number of transactions handled by the dm_oracle front-end process.	cm
brm_dmo_front_end_ops_done_total	Counter	The total number of operations handled by the dm_oracle front-end process.	cm
brm_dmo_back_end_ops_done_total	Counter	The total number of operations done by the dm_oracle back-end process.	cm
brm_dmo_back_end_ops_error_total	Counter	The total number of errors encountered by the dm_oracle back-end process.	cm
brm_dmo_back_end_trans_done_total	Counter	The total number of transactions handled by the dm_oracle back-end process.	cm
brm_dmo_back_end_trans_error_total	Counter	The total number of transaction errors in the dm_oracle back-end process.	cm
com_portal_js_JSMetrics_CurrentConnectionCount	Counter	The current concurrent connection to the Java Server from the CM.	eai_js
com_portal_js_JSMetrics_MaxConnectionCount	Counter	The maximum concurrent connections to the Java Server from the CM.	eai_js
com_portal_js_JSMetrics_SuccessfulOpcodeCount	Counter	The count of opcodes called from the CM, the execution of which succeeded in JS.	eai_js

Table 41-3 (Cont.) BRM Opcode Metrics

Metric Name	Metric Type	Description	BRM Service
com_portal_js_JSMetrics_FailedOpcodeCount	Counter	The count of opcodes called from the CM, the execution of which failed in JS.	eai_js
com_portal_js_JSMetrics_TotalOpcodeCount	Counter	The total count of opcodes called from the CM.	eai_js
com_portal_js_JSMetrics_TotalOpcodeExecutionTime	Counter	The total time taken in milliseconds across all opcodes.	eai_js

42
Monitoring Business Operations Center

Learn how to use external applications, such as Prometheus and Grafana, to monitor Oracle Communications Business Operations Center.

Topics in this document:

About Monitoring Business Operations Center

You set up the monitoring of Business Operations Center by using the following external applications:

- **WebLogic Monitoring Exporter**: Use this Oracle web application to scrape runtime information from Business Operations Center and then export the metric data into a format that can be processed by Prometheus. It exposes different WebLogic Mbeans metrics, such as memory usage and sessions count, that are required for monitoring and maintaining the Business Operations Center application.
- **Prometheus**: Use this open-source toolkit to aggregate and store the Business Operations Center metric data from WebLogic Monitoring Exporter.
- **Grafana**: Use this open-source tool to view on a graphical dashboard all Business Operations Center metric data stored in Prometheus.

Setting Up Monitoring in Business Operations Center

To set up monitoring in Business Operations Center:

1. Deploy a standalone version of Prometheus. See "Installation" in the Prometheus documentation.

   For the list of compatible software versions, see "Business Operations Center Software Compatibility" in *BRM Compatibility Matrix*.
2. Install Grafana. See "Install Grafana" in the Grafana documentation.

   For the list of compatible software versions, see "Business Operations Center Software Compatibility" in *BRM Compatibility Matrix*.
3. Configure WebLogic Monitoring Exporter to scrape metric data from Business Operations Center. See "Configuring WebLogic Monitoring Exporter for Business Operations Center".
4. Configure Prometheus to collect metric data. See "Configuring Prometheus for Business Operations Center".
5. Configure Grafana for displaying Business Operations Center metric data. See "Creating Grafana Dashboards for Business Operations Center".

Configuring WebLogic Monitoring Exporter for Business Operations Center

To configure WebLogic Monitoring Exporter to scrape metric data for Business Operations Center:

1. Download the latest supported version of WebLogic Monitoring Exporter (**get*N.N*.sh**) from https://github.com/oracle/weblogic-monitoring-exporter/releases, where *N.N* is the version number.

   For the list of supported versions, see "Business Operations Center Software Compatibility" in *BRM Compatibility Matrix*.

2. Edit the **wls-exporter-config.yaml** file to include the Business Operations Center metrics that you want scraped. For the list of supported metrics, see "WebLogic-Based Application Metrics".

3. Run the following command:

   ```
   bash getN.N.sh wls-exporter-config.yaml
   ```

   The **wls-exporter.war** is downloaded to your current directory.

4. Deploy the **wls-exporter.war** in the WebLogic Domain through the Admin Console, setting the target as all servers to be monitored.

WebLogic Monitoring Exporter is deployed on your servers and begins scraping metric data for Business Operations Center.

Example 42-1 Scraping Metrics for Business Operations Center

The following shows sample **wls-exporter-config.yaml** entries for setting up WebLogic Monitoring Exporter to scrape metrics from Business Operations Center:

```
metricsNameSnakeCase: true
queries:
- key: name
  keyName: location
  prefix: wls_server_
  applicationRuntimes:
    key: name
    keyName: app
    componentRuntimes:
      prefix: wls_webapp_config_
      type: WebAppComponentRuntime
      key: name
      values: [deploymentState, contextRoot, sourceInfo,
sessionsOpenedTotalCount, openSessionsCurrentCount,
openSessionsHighCount]
      servlets:
        prefix: wls_servlet_
        key: servletName
- JVMRuntime:
    prefix: wls_jvm_
    key: name
```

```
- executeQueueRuntimes:
    prefix: wls_socketmuxer_
    key: name
    values: [pendingRequestCurrentCount]
- workManagerRuntimes:
    prefix: wls_workmanager_
    key: name
    values: [stuckThreadCount, pendingRequests, completedRequests]
- threadPoolRuntime:
    prefix: wls_threadpool_
    key: name
    values: [executeThreadTotalCount, queueLength, stuckThreadCount,
hoggingThreadCount]
- JTARuntime:
    prefix: wls_jta_
    key: name
```

Configuring Prometheus for Business Operations Center

To configure a standalone version of Prometheus for Business Operations Center:

1. Configure Prometheus to collect your Business Operations Center metrics exposed by WebLogic Monitoring Exporter.

   To do so, add a scraping configuration to your **prometheus.yaml** file that is similar to the one shown below for each managed server and admin server in your cluster. In the following example, the Business Operations Center application is deployed in the **boc-domain** WebLogic domain and the server topology is:

   - The admin-server is on port 7001
   - A cluster named **cluster-1** with two managed servers
   - managed-server1 is on port 7003
   - managed-server2 is on port 7005

   ```
   scrape_configs:
     - job_name: 'wls-metric-boc'
       scrape_interval: 1s
       metrics_path: /wls-exporter/metrics
       scheme: http
       static_configs:
       - targets: ['localhost:7001']
         labels:
           weblogic_domainName: boc_domain
           weblogic_domainUID: boc_domain
           weblogic_serverName: admin-server
       - targets: ['localhost:7003']
         labels:
           weblogic_clusterName: cluster-1
           weblogic_domainName: boc_domain
           weblogic_domainUID: boc_domain
           weblogic_serverName: managed-server1
       - targets: ['localhost:7005']
         labels:
           weblogic_clusterName: cluster-1
   ```

```
        weblogic_domainName: boc_domain
        weblogic_domainUID: boc_domain
        weblogic_serverName: managed-server2
    basic_auth:
      username: username
      password: password
```

where *username* and *password* is your WebLogic Server user name and password.

2. Configure the alert rules in Prometheus.

   To do so, add alert rules that are similar to the ones shown below to your **prometheus-rules.yaml** file. For examples of expressions that you can use for memory and CPU usage, see "Sample Expressions for CPU and Memory Usage".

```
groups:
  - name: alertmanager.rules
    rules:
      - alert: ClusterWarning
        annotations:
          message: WLS cluster has less than 2 managed servers
running for more than 1 minute.
        expr: sum by(weblogic_domainUID, weblogic_clusterName,
weblogic_serverName) (up{weblogic_domainUID=~'.+'}) < 2
        for: 1m
        labels:
          severity: critical
```

3. Configure Prometheus Alertmanager to raise alerts.

   To do so, add alert configurations that are similar to the ones shown below to your **prometheus.yaml** file.

```
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
       - localhost:9093
# Load rules once and periodically evaluate them according to the
global 'evaluation_interval'.
rule_files:
  - "prometheus-rules.yaml"
```

4. Stop and restart Prometheus.

Creating Grafana Dashboards for Business Operations Center

Create a dashboard in Grafana for displaying your Business Operations Center metric data. You can alternatively use the sample dashboard JSON model that is included with the Business Operations Center package.

> **Note:**
>
> For the sample dashboard to work properly, the datasource name for the WebLogic Domain must be **Prometheus**.

To use the sample dashboard, import the *Domain_home***/samples/monitoring/boc-dashboard.json** dashboard file into Grafana, where *Domain_home* is the WebLogic Server domain home directory in which Business Operations Center is deployed. For information about importing dashboards into Grafana, see "Export and Import" in the *Grafana Dashboards* documentation.

Sample Expressions for CPU and Memory Usage

You can use the following sample expressions in your Prometheus alert rules that are based on current CPU and memory usage.

To raise an alert when the average CPU usage across all managed servers is greater than 70% for more that two minutes:

```
avg(avg_over_time(wls_jvm_process_cpu_load{weblogic_domainUID="boc-
domain",weblogic_serverName=~"managed-server.*"}[2m]))*100 > 70
```

To raise an alert when the average memory usage across all managed servers is greater than 70% for more than two minutes:

```
100 - avg(avg_over_time(wls_jvm_heap_free_percent{weblogic_domainUID="app-
domain",weblogic_serverName=~"managed-server.*"}[2m])) > 70
```

43

Monitoring Billing Care and Billing Care REST API

Learn how to use external applications, such as Prometheus and Grafana, to monitor Oracle Communications Billing Care and Oracle Communications Billing Care REST API.

Topics in this document:

About Monitoring Billing Care and Billing Care REST API

You set up the monitoring of Billing Care and the Billing Care REST API by using the following external applications:

- **WebLogic Monitoring Exporter**: Use this Oracle web application to scrape runtime information from Billing Care and the Billing Care REST API, and then export the metric data into a format that can be processed by Prometheus. It exposes different WebLogic Mbeans metrics, such as memory usage and sessions count, that are required for monitoring and maintaining the application.
- **Prometheus**: Use this open-source toolkit to aggregate and store the metric data from WebLogic Monitoring Exporter.
- **Grafana**: Use this open-source tool to view on a graphical dashboard all Billing Care and Billing Care REST API metric data stored in Prometheus.

Setting Up Monitoring in Billing Care and Billing Care REST API

To set up monitoring in Billing Care and the Billing Care REST API:

1. Install the following external software on your system:
   - Pometheus. See "Installation" in the Prometheus documentation.
   - Grafana. See "Install Grafana" in the Grafana documentation.

   For the list of compatible software versions, see "Billing Care Software Compatibility" in *BRM Compatibility Matrix*.
2. Configure WebLogic Monitoring Exporter to scrape metric data from Billing Care and the Billing Care REST API. See "Configuring WebLogic Monitoring Exporter for Billing Care and Billing Care REST API".
3. Configure Prometheus to scrape metric data from WebLogic Monitoring Exporter. See "Configuring Prometheus for Billing Care and Billing Care REST API".

4. Configure Grafana to display your Billing Care and Billing Care REST API metric data in a graphical format. See "Creating Grafana Dashboards for Billing Care and Billing Care REST API".

Configuring WebLogic Monitoring Exporter for Billing Care and Billing Care REST API

To configure WebLogic Monitoring Exporter to scrape metric data for Billing Care and the Billing Care REST API:

1. Download the latest supported version of WebLogic Monitoring Exporter (**get*N.N*.sh**) from https://github.com/oracle/weblogic-monitoring-exporter/releases, where *N.N* is the version number.

   For the list of supported versions, see "Billing Care Software Compatibility" in *BRM Compatibility Matrix*.

2. Edit the **wls-exporter-config.yaml** file to include the metrics that you want scraped. For the list of supported metrics, see "WebLogic-Based Application Metrics".

3. Run the following command:

   ```
   bash getN.N.sh wls-exporter-config.yaml
   ```

   The **wls-exporter.war** is downloaded to your current directory.

4. Deploy the **wls-exporter.war** in the WebLogic Domain through the Admin Console, setting the target as all servers to be monitored.

5. Verify that metrics are getting exported by running the following cURL command:

   ```
   curl -X GET -i -u WL_userName:WL_password http://localhost:8080/
   metrics
   ```

   where *WL_userName* and *WL_password* is the user name and password for WebLogic Server.

WebLogic Monitoring Exporter is deployed on your servers and begins scraping metric data for Billing Care and the Billing Care REST API.

Example 43-1 Configuring WebLogic Monitoring Exporter to Scrape Metrics

The following shows sample **wls-exporter-config.yaml** entries for setting up WebLogic Monitoring Exporter to scrape metrics from Billing Care and Billing Care REST API:

```
metricsNameSnakeCase: true
queries:
- key: name
  keyName: location
  prefix: wls_server_
  applicationRuntimes:
    key: name
    keyName: app
    componentRuntimes:
```

```
      prefix: wls_webapp_config_
      type: WebAppComponentRuntime
      key: name
      values: [deploymentState, contextRoot, sourceInfo,
sessionsOpenedTotalCount, openSessionsCurrentCount, openSessionsHighCount]
      servlets:
        prefix: wls_servlet_
        key: servletName
- JVMRuntime:
    prefix: wls_jvm_
    key: name
- executeQueueRuntimes:
    prefix: wls_socketmuxer_
    key: name
    values: [pendingRequestCurrentCount]
- workManagerRuntimes:
    prefix: wls_workmanager_
    key: name
    values: [stuckThreadCount, pendingRequests, completedRequests]
- threadPoolRuntime:
    prefix: wls_threadpool_
    key: name
    values: [executeThreadTotalCount, queueLength, stuckThreadCount,
hoggingThreadCount]
- JTARuntime:
    prefix: wls_jta_
    key: name
```

Configuring Prometheus for Billing Care and Billing Care REST API

To configure a standalone version of Prometheus for Billing Care and the Billing Care REST API:

1. Configure Prometheus to scrape the required metrics exposed by WebLogic Monitoring Exporter.

   To do so, add a scraping configuration to your **prometheus.yaml** file that is similar to the one shown below for each managed server and admin server in your cluster. In the following example, the Billing Care application is deployed in the **billingcare-domain** WebLogic domain and the server topology is:

   - The admin-server is on port 7001
   - A cluster named **cluster-1** with two managed servers
   - managed-server1 is on port 7003
   - managed-server2 is on port 7005

   The Billing Care REST API is deployed in the **bcws-domain** WebLogic domain and the server topology is:

   - The admin-server is on port 7011
   - A cluster named **cluster-1** with two managed servers
   - managed-server1 is on port 7013

- managed-server2 is on port 7015

```
scrape_configs:
  - job_name: 'wls-metric-billingcare'
    scrape_interval: 1s
    metrics_path: /wls-exporter/metrics
    scheme: http
    static_configs:
    - targets: ['localhost:7001']
      labels:
        weblogic_domainName: billingcare_domain
        weblogic_domainUID: billingcare_domain
        weblogic_serverName: admin-server
    - targets: ['localhost:7003']
      labels:
        weblogic_clusterName: cluster-1
        weblogic_domainName: billingcare_domain
        weblogic_domainUID: billingcare_domain
        weblogic_serverName: managed-server1
    - targets: ['localhost:7005']
      labels:
        weblogic_clusterName: cluster-1
        weblogic_domainName: billingcare_domain
        weblogic_domainUID: billingcare_domain
        weblogic_serverName: managed-server2
    basic_auth:
      username: <username>
      password: <password>
  - job_name: 'wls-metric-bcws'
    scrape_interval: 1s
    metrics_path: /wls-exporter/metrics
    scheme: http
    static_configs:
    - targets: ['localhost:7011']
      labels:
        weblogic_domainName: bcws_domain
        weblogic_domainUID: bcws_domain
        weblogic_serverName: admin-server
    - targets: ['localhost:7013']
      labels:
        weblogic_clusterName: cluster-1
        weblogic_domainName: bcws_domain
        weblogic_domainUID: bcws_domain
        weblogic_serverName: managed-server1
    - targets: ['localhost:7015']
      labels:
        weblogic_clusterName: cluster-1
        weblogic_domainName: bcws_domain
        weblogic_domainUID: bcws_domain
        weblogic_serverName: managed-server2
    basic_auth:
     username: username
     password: password
```

where *username* and *password* is your WebLogic Server user name and password.

2. Configure the alert rules in Prometheus.

   To do so, add alert rules that are similar to the ones shown below to your **prometheus-rules.yaml** file. For examples of expressions that you can use for memory and CPU usage, see "Sample Expressions for CPU and Memory Usage".

```
groups:
  - name: alertmanager.rules
    rules:
      - alert: ClusterWarning
        annotations:
          message: WLS cluster has less than 2 running server for more
than 1 minutes.
        expr: sum by(weblogic_domainUID, weblogic_clusterName,
weblogic_serverName) (up{weblogic_domainUID="billingcare-domain"}) <
2
        for: 1m
        labels:
          severity: critical
      - alert: ClusterWarning
        annotations:
          message: WLS cluster has less than 2 running server for more
than 1 minutes.
        expr: sum by(weblogic_domainUID, weblogic_clusterName,
weblogic_serverName) (up{weblogic_domainUID="bcws-domain"}) < 2
        for: 1m
        labels:
          severity: critical
```

3. Configure Prometheus Alertmanger to raise alerts.

   To do so, add an alert configuration to your **prometheus.yaml** file that is similar to the one shown below:

```
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
       - localhost:9093
# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
  - "prometheus-rules.yaml"
```

4. Stop and restart Prometheus.

Creating Grafana Dashboards for Billing Care and Billing Care REST API

You can create a dashboard in Grafana for displaying your Billing Care and Billing Care REST API metric data.

Alternatively, you can use the sample dashboards that are included with the Billing Care and Billing Care SDK packages. To use the sample dashboard, import the following dashboard

files into Grafana. For information about importing dashboards into Grafana, see "Export and Import" in the *Grafana Dashboards* documentation.

- Billing Care: *Domain_home*/**samples/monitoring/billingcare-dashboard.json**
- Billing Care REST API: *SDK_home*/**BillingCareREST/samples/monitoring/billingcare-rest-api-dashboard.json**

where *Domain_home* is the WebLogic Server domain home directory in which Billing Care is deployed, and *SDK_home* is the directory in which you installed the Billing Care SDK.

> **Note:**
>
> For the sample dashboard to work properly, the datasource name for the WebLogic Domain must be **Prometheus**.

Sample Expressions for CPU and Memory Usage

You can use the following sample expressions in your Prometheus alert rules that are based on current CPU and memory usage.

To raise an alert when the average CPU usage across all managed servers is greater than 70% for more that two minutes:

- For the Billing Care REST API domain:

```
avg(avg_over_time(wls_jvm_process_cpu_load{weblogic_domainUID="bcws-
domain",weblogic_serverName=~"managed-server.*"}[2m]))*100 > 70
```

- For the Billing Care domain:

```
avg(avg_over_time(wls_jvm_process_cpu_load{weblogic_domainUID="billi
ngcare-domain",weblogic_serverName=~"managed-server.*"}[2m]))*100 >
70
```

To raise an alert when the average memory usage across all managed servers is greater than 70% for more than two minutes:

- For the Billing Care REST API domain:

```
100 -
avg(avg_over_time(wls_jvm_heap_free_percent{weblogic_domainUID="bcws
-domain",weblogic_serverName=~"managed-server.*"}[2m])) > 70
```

- For the Billing Care domain:

```
100 -
avg(avg_over_time(wls_jvm_heap_free_percent{weblogic_domainUID="bill
ingcare-domain",weblogic_serverName=~"managed-server.*"}[2m])) > 70
```

Part VI

Partitioning and Managing BRM Database Tables

This part describes how to partition the Oracle Communications Billing and Revenue Management (BRM) database, how to purge data, and how to generate virtual columns on event tables. It contains the following chapters:

44
Partitioning Tables

Learn how to organize your Oracle Communications Billing and Revenue Management (BRM) database by using partitioned tables.

Topics in this document:

Before partitioning any tables, you should be familiar with the following:

- Basic BRM concepts. See "About BRM" in *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- Oracle documentation and Oracle database administration skills, including PL/SQL and SQL*Plus.
- Perl.
- UNIX commands and the UNIX operating system.
- The **pin_setup.values** file for configuring the BRM server. See *BRM Installation Guide*.

About Partitioning

Partitioning splits tables and indexes into smaller, more manageable units. When you no longer need the data stored in a partition, you can purge it from your database by deleting the partition (see "About Purging Data").

To partition tables, you need Oracle Partitioning. You license this third-party software from Oracle.

You can enable partitioning at the following times:

- When installing BRM. See "Installing BRM" in *BRM Installation Guide*.
- After installing BRM. See "Converting Nonpartitioned Classes to Partitioned Classes".

If you enable partitioning during installation, the following storable classes and their subclasses are automatically enabled for partitioning:

- event
- bill
- invoice
- item
- journal
- newsfeed
- sepa
- user activity

To enable partitioning for a different set of storable classes after installing BRM, see "Converting Nonpartitioned Classes to Partitioned Classes".

When you use partitioning, objects are stored in a partition according to the following criteria:

- Nonpurgeable event and item objects are stored in **partition_historic**. See "About Nonpurgeable Events and Items".
- All other objects are stored in date-based, purgeable partitions according to the date they were created. For example, if a partition includes a date range from January 15 to February 15, all objects that have a creation date in that range are included.

  You create separate partitions for real-time events and delayed events. You can purge real-time events, delayed events, or both. See "About Partitions for Delayed Events".

  > **Note:**
  >
  > Delayed partitioning is not supported for non-event storable classes.

- If no date-based partition exists, objects are stored in **partition_last**. See "About Objects Stored in partition_last and partition_last_pin".

For partitioned event tables, you can use the default purgeable partitions that **pin_setup** creates, or you can create custom purgeable partitions. For non-event tables, you must create custom purgeable partitions.

See "About Partitioning Schemes" and "Customizing Partition Limitations".

> **Note:**
>
> A partition applies to all the tables for a base storable class and its subclasses. When you remove a partition, it removes partitions in all the partitioned tables for that base storable class and its subclasses.

To create partitions and purge data, see "About Managing Partitions".

> **Note:**
>
> Rule-based optimization is not available for partitioned tables and indexes. See "Using Rule-Based Optimization versus Cost-Based Optimization" in *BRM Installation Guide*.

About Partitioning Schemes

Partitions are based on dates; your partitioning scheme determines how the date-based time periods are divided. Partition time periods can be daily, weekly, or monthly. You can specify multiples of each time period; for example, you can create partitions based on five days or on three weeks.

Determining the appropriate partition size depends on many factors. Generally, partitions should not exceed 120,000,000 entries for standard BRM configurations. For more information, see the Oracle database documentation, or contact Oracle support.

You can set up partitioning schemes based on different time periods in advance; for example, you can specify three monthly partitions followed by three weekly partitions. However, every table of the same base storable class must have the same partitioning scheme; you cannot create different partitions for different tables based on the same base storable class.

> **Note:**
>
> When you add partitions, you define a start date for the first partition. The start date cannot be earlier than the day after tomorrow. For example, if the current date is January 1, the earliest start date for the new partition is January 3.

Rated Event (RE) Loader loads delayed events into the BRM database. You can enable partitioning for delayed events and create partitions for the delayed events. See "About Partitions for Delayed Events".

> **Note:**
>
> When you partition an event table, the associated index is identically partitioned by the Oracle database.

About Nonpurgeable Events and Items

By default, certain storable classes of events cannot be purged from the database. In addition, you can make other storable classes of events and items nonpurgeable.

All event and item objects belonging to nonpurgeable storable classes are automatically stored in **partition_historic**. You cannot remove this partition.

For the default list of events stored in **partition_historic**, see "Event and Item Objects Stored in partition_historic".

To modify the list of events stored in **partition_historic**, see "Customizing the List of Events and Items Stored in partition_historic".

Associating Items with Nonpurgeable Events

In BRM, events and items are mapped to one another in item configuration objects (***/config/item_tags*** and ***/config/item_types***). If an event in an item configuration is nonpurgeable, the items mapped to it should also be nonpurgeable because the event is incomplete without those items.

To ensure that nonpurgeable events are not associated with purgeable items, do one of the following:

- Synchronizing the Purgeability of Events and Items in New Installations
- Handling Items Associated with Nonpurgeable and Purgeable Events in Existing Systems

Synchronizing the Purgeability of Events and Items in New Installations

If you enabled partitioning for item storable classes when you installed BRM, *before* your system begins generating objects, synchronize the purgeable/nonpurgeable status of associated event and item storable classes.

To synchronize the purgeability of events and items in new BRM installations:

1. Make a list of the event storable classes that are nonpurgeable by default.

   To determine whether an event storable class is nonpurgeable, consult its specification. See the discussion on retrieving storable class specifications in *BRM Configuring and Running Billing*.

2. Make a list of the item storable classes associated with the nonpurgeable events listed in the previous step.

   To identify such classes, consult the following files in *BRM_home****/sys/data/pricing/example***:

   - **config_item_tags.xml**
   - **config_item_types.xml**

   For information about how events and items are associated in those files, see "Creating Custom Bill Items" in *BRM Configuring and Running Billing*.

3. Verify that all item types associated with nonpurgeable events are also nonpurgeable.

4. If you find purgeable item types associated with nonpurgeable events, make such item types nonpurgeable.

   For more information, see the Storable Class Editor Help.

5. If you create any custom item subclasses that will be associated with nonpurgeable events, select the subclass's **Non-Purgable** check box.

   For more information about storable class properties, see the Storable Class Editor Help.

Handling Items Associated with Nonpurgeable and Purgeable Events in Existing Systems

If you convert a nonpartitioned item storable class to a partitioned storable class in an existing BRM system, the system is likely to contain item objects associated with both purgeable and nonpurgeable events. Problems can occur if such items are purged. To minimize such problems, do the following *before* converting the nonpartitioned item storable class to a partitioned storable class:

To handle items associated with both nonpurgeable and purgeable events in existing BRM systems:

1. Make a list of the event storable classes that are nonpurgeable by default.

   To determine whether an event storable class is nonpurgeable, consult its specification. See the discussion on retrieving storable class specifications in *BRM Developer's Guide*.

2. Make a list of the item storable classes associated with the nonpurgeable events listed in the previous step.

   To identify such classes, consult the following files in *BRM_home***/sys/data/pricing/example**:

   - **config_item_tags.xml**
   - **config_item_types.xml**

   For information about how events and items are associated in those files, see "Creating Custom Bill Items" in *BRM Configuring and Running Billing*.

3. Check whether any of the item storable classes associated with nonpurgeable events are also associated with purgeable events.

   To identify such classes, consult the configuration files listed in the previous step.

4. For item storable classes that are associated with both nonpurgeable and purgeable events, consider doing one or more of the following:

   - Split the item storable class into two classes, one for nonpurgeable events and one for purgeable events. Update the item configuration files accordingly.

     Set the **Non-Purgable** property correctly for each storable class. (See the Storable Class Editor Help.)

     This may add a line to your invoices. It may also affect custom code.

   - Make the nonpurgeable events associated with such items purgeable.

     Ensure that the item storable class is also purgeable.

   - Make the item storable class nonpurgeable.

     The item objects will not be purged even when associated with purgeable events, so the storage requirements for this item storable class will continue to grow and may affect performance.

   - Make the item storable class purgeable.

     The links between purged item objects and nonpurgeable events will be broken. Future operations such as event adjustments or cancellations may result in errors.

5. Convert the nonpartitioned item storable class to a partitioned storable class.

   See "Converting Nonpartitioned Classes to Partitioned Classes".

About Objects Stored in partition_last and partition_last_pin

The **partition_last** partition is a spillover partition that holds objects dated after the time period covered by the regular partitions if no additional purgeable partitions are available for such objects. You cannot remove this partition for real-time event tables or non-event tables.

If your partitions are enabled for delayed events, **partition_last** is used for delayed events, and **partition_last_pin** is used for real-time events.

> **Note:**
>
> Spillover partitions are not intended to store objects you want to purge or preserve; they are just temporary containers. To keep objects out of spillover partitions, make sure your tables contain partitions to hold new objects. See "Adding Partitions".

To purge objects stored in spillover partitions, stop all delayed-event loading, and then add partitions by using the **partition_utils** utility with the **-f** parameter (see "Adding Partitions"). Adding partitions in this way moves the data from the spillover partitions to the added partitions, but it takes much longer than adding partitions normally.

About the Default Partitioning Scheme

If you enable partitioning when you install BRM and choose to create the default partitions, your real-time events are stored in the following partitions:

- **partition_historic** (event and item tables only)
- **partition_last**
- 12 monthly partitions
- +1 partition for objects created on the start date

Each monthly partition stores objects created on the date of the month you install BRM through the previous date of the next month. For example, if you install BRM on January 15, the partitions cover January 15 through February 14, February 15 through March 14, and so on.

For information about partition names, see "Partition Naming Convention".

If You Do Not Create Default Partitions

If you enable partitioning when you install BRM but choose *not* to create the default partitions, your real-time object tables are divided into fewer partitions: **partition_historic** (stores all nonpurgeable events and items) and **partition_last** (stores all purgeable objects). This is sufficient for a test or development system in which purging by partition is not required. For a production system, however, you must add more purgeable partitions. See "Adding Partitions".

Conversion Partitioning Scheme

If you enable partitioning by converting nonpartitioned storable classes to partitioned storable classes *after* you install BRM, your partitioned real-time tables are divided into three partitions:

- **partition_migrate**: Holds all event objects created before the conversion. The **partition_utils** utility cannot purge objects in this partition. To purge them, you must develop your own tools based on sound Oracle database management principles.
- **partition_historic**: Holds nonpurgeable events and items created after the conversion. Nonpurgeable events and items should not be purged from the database. See "Event and Item Objects Stored in partition_historic".
- **partition_last**: A spillover partition that is not intended to store event objects you want to purge or preserve. If you do not add purgeable partitions to your tables before BRM resumes generating objects, purgeable objects created after the conversion are stored in this partition. See "About Objects Stored in partition_last and partition_last_pin".

About Partitions for Delayed Events

To create partitions for delayed events, use the **partition_utils** utility to enable delayed-event partitions for specific event storable classes. See "Enabling Delayed Partitions".

When you enable delayed-event partitions, **partition_utils** automatically runs the **update** operation. This adds partitions for the delayed events that are aligned with any real-time event partitions already defined in the EVENT_T table. Therefore, you do not need to explicitly add partitions for delayed events after enabling them.

To use a partitioning scheme for delayed events that differs from that already defined in the EVENT_T table, you must explicitly add delayed-event partitions. You add delayed-event partitions the same way you add real-time partitions.

When partitions are added for delayed events, the event tables are divided into the default partitions plus the following partitions:

- **Partitions for delayed events:** The time periods covered by these partitions do not need to match the periods covered by the partitions for real-time event objects. For example, you can use daily partitions for real-time events and weekly partitions for delayed events.
- **partition_last_pin**: When delayed-event partitions are added to tables, this spillover partition is added for *real-time* event objects; **partition_last** becomes the spillover partition for delayed events for which no other partition is available. See "About Objects Stored in partition_last and partition_last_pin".

Delayed-event storage is based on the date that the delayed events are loaded into the BRM database.

If you enable delayed-event partitions for an event storable class but then decide you do not need those partitions, you can use **sqlplus** to remove the delayed-event partitions and then to remove the **partition_last_pin** partition. See "Disabling Delayed-Event Partitioning".

Overview of Partitioning Schemes

If you do not enable partitions when installing BRM, every object table has no partitions.

The following tables show the possible partitioning schemes.

An object table designated for partitioning is partitioned as shown in Table 44-1 if you enable partitions but do not create the 12 default partitions.

Table 44-1 Real-Time Partitioning Enabled, No Default Partitions

Partition	Partition
partition_historic (event and item tables only)	**partition_last**

> **Note:**
>
> Do not use the partitioning scheme shown in Table 44-1 in a production system because all non-historic real-time objects are stored in **partition_last**. You must add real-time partitions.

An object table designated for partitioning is partitioned as shown in Table 44-2 if you enable partitions and accept the 12 default partitions.

Table 44-2 Real-Time Partitioning Enabled, 12 Default Partitions

Partition	Partition	Partition	Partition	Partition	Partition	Partition
partition_historic (event and item tables only)	Real Time Partition 1	Real Time Partition 2	Real Time Partitions 3 through 10	Real Time Partition 11	Real Time Partition 12	**partition_last**

An object table designated for partitioning is partitioned as shown in Table 44-3 if you enable partitions and create custom partitions.

Table 44-3 Real-Time Partitioning Enabled, Custom Default Partitions

Partition	Partition	Partition	Partition	Partition	Partition	Partition
partition_historic (event and item tables only)	Real Time Partition 1	Real Time Partition 2	Real Time Partition 3	Real Time Partitions 4 through *N - 1*	Real Time Partition *N*	**partition_last**

An event table is partitioned as shown in Table 44-4 if you enable partitions without the 12 default partitions and then enable delayed-event partitions.

Table 44-4 Real-Time Partitioning Enabled, Delayed Partitioning Enabled, No Default Partitions

Partition	Partition	Partition
partition_historic	**partition_last_pin**	**partition_last**

> **Note:**
>
> Do not use the partitioning scheme shown in Table 44-4 in a production system because all non-historic events are stored in **partition_last** and **partition_last_pin**. You must add real-time and delayed partitions.

An event table is partitioned as shown in Table 44-5 if you enable partitions, create real-time partitions, and enable delayed-event partitions.

Table 44-5 Real-Time Partitioning Enabled, Real-Time Partitions Exist, Delayed Partitioning Enabled

Partition	Partition	Partition	Partition	Partition	Partition	Partition	Partition
partition_historic	RT P_1	RT P_2	RT P_3	RT P_4 through RT P_(*N*-1)	RT P_*N*	**partition_last_pin**	**partition_last**

> **Note:**
>
> Do not use the partitioning scheme shown in Table 44-5 in a production system because all non-historic delayed events are stored in **partition_last**. You must add delayed partitions.

An event table is partitioned as shown in Table 44-6 if you enable partitions, do not create real-time partitions, and enable and create delayed partitions.

Table 44-6 Real-Time Partitioning Enabled, No Real-Time Partitions, Delayed Partitioning Enabled, Delayed Partitions Exist

Partition	Partition	Partition	Partition	Partition	Partition	Partition	Partition
partition_historic	**partition_last_pin**	D P_1	D P_2	D P_3	D P3 through D P_(*N*-1)	D P_*N*	**partition_last**

> **Note:**
>
> Do not use the partitioning scheme shown in Table 44-6 in a production system because all non-historic real-time events are stored in **partition_last_pin**. You must add real-time partitions.

An event table is partitioned as shown in Table 44-7 if you enable partitions, create real-time partitions, and enable and create delayed partitions.

Table 44-7 Real-Time Partitioning Enabled, Real-Time Partitions Exist, Delayed Partitioning Enabled, Delayed Partitions Exist

Partition	Part'n	Part'n	Part'n	Part'n	Part'n	Part'n	Part'n	Part'n	Part'n	Part'n
partition_historic	RT P_1	RT P_2	RT P_3 through RT P_(*N*-1)	RT P_*N*	**partition_last_pin**	D P_1	D P_2	D P_3 through D P_(*N*-1)	D P_*N*	**partition_last**

About Managing Partitions

To manage partitions, you perform the following tasks:

- Customize partitions before you use a production system. See "Adding Partitions".
- Enable delayed partitions. You can specify which delayed events you want to create partitions for. See "Enabling Delayed-Event Partitioning" and "Adding Partitions".
- Update partitions when you customize storable classes with partitioned tables or add storable classes by adding components. See "Updating Partitions".
- Purge objects by removing partitions. See "Purging Objects by Removing Partitions".
- Purge objects by purging old objects without removing partitions. See "Purging Objects without Removing Partitions".
- Add real-time partitions or delayed-event partitions to store new objects. See "Adding Partitions".
- Customize the events and items stored in **partition_historic**. See "Customizing the List of Events and Items Stored in partition_historic".

Before your BRM production system begins generating objects that you want to store in partitions, add partitions to your tables if any of the following situations is true:

- (Event tables only) You removed the 12 default monthly event partitions created during installation.
- (Event tables only) You chose not to create the 12 default monthly event partitions when you enabled partitioning during installation.
- You partitioned your tables by upgrading your database.
- You enabled partitioning for any non-event storable classes.

In addition, after your partitioned production system is running, you should add purgeable partitions whenever the time periods covered by the existing purgeable partitions are about to pass. See "About Objects Stored in partition_last and partition_last_pin".

About Purging Objects

When you purge objects, you can do one of the following:

- Remove the partitions that contain the objects. See "About Purging Objects by Removing Partitions".

- Purge the objects, but retain the object partitions. See "About Purging Objects without Removing Partitions".

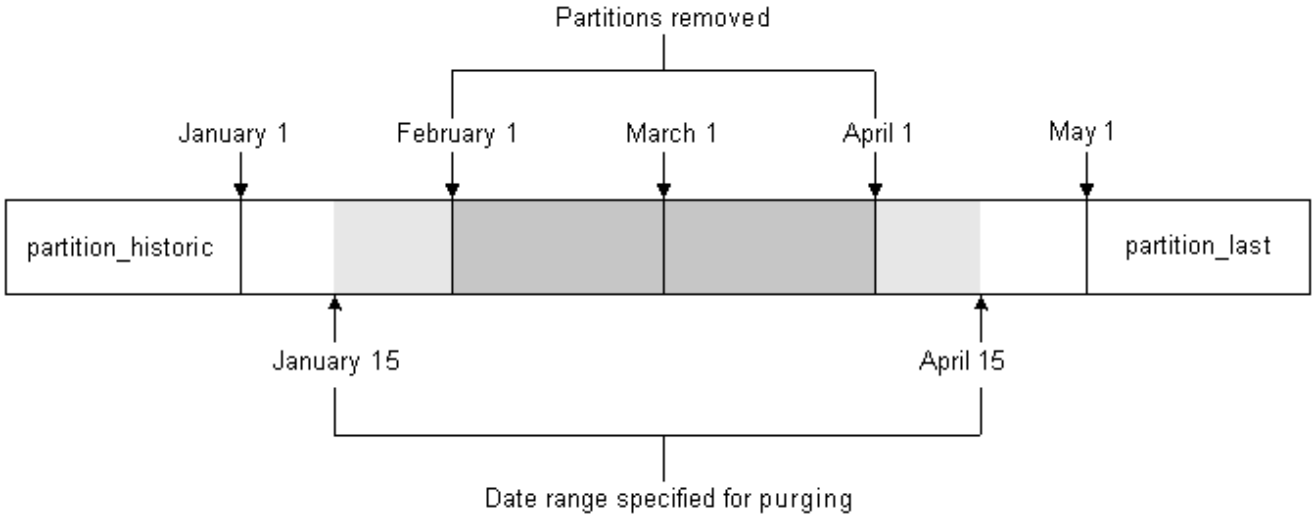
About Purging Objects by Removing Partitions

When you purge objects by removing their partitions, you specify a start date and an end date to identify the objects to be removed. For example, you can purge objects created from January 15 to March 15.

> **Note:**
>
> - The dates you specify for purging objects might not align with the partition dates. In this case, only partitions completely within the date range are removed.
> - If a partition within the date range contains an event associated with an open item, by default, the partition is not removed. To change the default behavior, see "Enabling Open Items to Be Purged".
> - An object's oldest partition is not purged by default. To force the removal of the oldest partition, you must set the start date for purging objects to January 1, 1970 (that is, **-s 01011970**).

Figure 44-1 shows monthly partitions, each starting on the first of the month. If you specify to purge objects created between January 15 and April 15, only objects from February 1 through April 1 are purged because those are the only objects in partitions completely within the date range.

Figure 44-1 Purging of Complete Partitions



For more information, see "Purging Objects by Removing Partitions".

> **Note:**
>
> Purging partitions is considered safer than removing them. For more information, see "About Purging Objects without Removing Partitions".

> **Tip:**
>
> Before purging objects, run the **partition_utils** utility with the **-p** parameter to write an SQL statement that shows the partitions that will be removed. See "Running the partition_utils Utility in Test Mode".

About Purging Objects without Removing Partitions

You can delete old objects without deleting partitions by specifying the last object date to keep. If any old events are associated with active items, those particular objects are not deleted. See "Purging Objects without Removing Partitions".

> **Tip:**
>
> Before purging objects, run the **partition_utils** utility with the **-p** parameter to write an SQL statement that shows the partitions that will be removed. See "Running the partition_utils Utility in Test Mode".

> **Note:**
>
> Purging partitions is considered safer than removing them.

About Running the partition_utils Utility

Most partition management tasks are accomplished by running the **partition_utils** utility.

> **Note:**
>
> After using your most current version of the **partition_utils** utility, *do not* use previous versions. Doing so can corrupt your database. (Partitions created by using the previous version are, however, supported).

You can run only one instance of this utility at a time. If you try to run more than one instance, the utility does not run and returns an error.

> **Note:**
>
> After starting the utility, do not interrupt it. It might take several minutes to finish, depending on the size of your database. If the utility is interrupted, use the **partition_utils restart** operation to continue the previous operation. See "Restarting partition_utils".

The **partition_utils** utility creates the following tables. Do not delete them or use the table names for any other purpose:

- PIN_TEMP_BRM_SERVER_TIME_T
- PIN_TEMP_DD_INFO_T
- PIN_TEMP_DETAILS_T
- PIN_TEMP_HIGH_VAL_T
- PIN_TEMP_OUT_OF_SYNC_T
- PIN_TEMP_PURGE_LOGIC_T
- PIN_TEMP_PURGE_PARTITIONS_T
- PIN_TEMP_PURGE_POIDS_T
- PIN_TEMP_PURGE_STATISTICS_T
- PIN_TEMP_TAB_PARTITIONS_T
- PIN_TEMP_TBL_NAMES_T

Partition Naming Convention

The naming convention for real-time partitions is the following:

P_R_*MMDDYYYY*

For example, the following partition name specifies that the last date used for objects is June 29, 2004:

```
P_R_06292004
```

The naming convention for delayed-event partitions is the following:

P_D_*MMDDYYYY*

For example, the following partition name specifies that the last date used for objects is June 29, 2004:

```
P_D_06292004
```

Running the partition_utils Utility in Test Mode

When you add, remove, update, or enable partitions, you can use the **-p** parameter to write a SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. The **partition_utils.log** file is in the same directory as the **partition_utils** utility.

> **Note:**
>
> Do not copy the SQL statements and run them against the database. This action is not supported.

Configuring a Database Connection

Before you use the **partition_utils** utility, configure the database connection parameters in the **partition_utils.values** file in *BRM_home*/**apps/partition_utils**. For example:

```
$MAIN_DB{'vendor'}   =  "oracle";
$MAIN_DB{'alias'}    =  "pindb.example.com";
$MAIN_DB{'user'}     =  "pin";
$MAIN_DB{'password'} =  "password";
```

For more information, see the comments in the **partition_utils** file.

Improving Performance When Using partition_utils

For all **partition_utils** operations, you can run processes in parallel to improve performance.

To run processes in parallel, edit the NUM_OF_PROCESSES parameter in the *BRM_home*/**apps/partition_utils/partition_utils.values** file. This parameter controls how many **partition_utils** processes can run in parallel, except when the **enable** operation is run.

The valid range for NUM_OF_PROCESSES is from **1** to **10**. The default is **2**.

Restarting partition_utils

If **partition_utils** is interrupted, it is best to use **partition_utils** with the **restart** operation to continue the previous operation. This prevents your database partitions from becoming unaligned and corrupted.

The syntax is the following:

```
partition_utils -o restart [-b]
```

Use the **-b** (bypass) parameter to bypass the last operation and clean the status of it.

For more information, see "partition_utils".

Adding Partitions

To add partitions, run the **partition_utils** utility with the **add** operation.

> **Note:**
>
> - Partitioning must be enabled in your system before you can add partitions. If partitioning is not enabled, see "Converting Nonpartitioned Classes to Partitioned Classes".
> - Delayed-event partitions must be enabled before you can add delayed-event partitions. See "Enabling Delayed-Event Partitioning".
> - To use this utility to add a partition for a storable class other than the event, bill, invoice, item, journal, newsfeed, sepa, and user activity storable classes, partitioning must have been enabled for that storable class in one of the following ways:
>
>   – By editing the **pin_setup.values** file during the BRM installation process. See “Enabling Different Classes for Partitioning during Installation" in *BRM Installation Guide*.
>
>   – By using the partitioning package to convert the storable class after BRM is installed. See "Converting Nonpartitioned Classes to Partitioned Classes" in *BRM System Administrator's Guide*.
> - When partitions are added, an additional partition for the time up to *start_date* is created by default if no partition for that time exists.

The syntax is the following:

```
partition_utils   -o add -t realtime|delayed -s start_date
                  -u month|week|day -q quantity
                  [-c storable_class] [-w width]
                  [-f] [-p]
```

- Use the **-t** (type) parameter to add real-time or delayed-event partitions. Only event tables can have delayed-event partitions.

  > **Note:**
  >
  > Non-event partitions are always real-time. You can add only delayed-event partitions. To do so, you must enable partitioning for each event type you want to partition. See "Enabling Delayed-Event Partitioning".

- Use the **-s** (start date) parameter to specify the start date for the first of the new partitions. The format is *MMDDYYYY*.

  The start date must be the day after tomorrow or later; you cannot create partitions starting on the current day or the next day. For example, if the current date is January 1, the earliest start date for the new partition is January 3.

  By default, the start date must also be earlier than six months from today. You can change these defaults by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".

  If you try to create a partition with a start date within the current partition, the utility reports an error and provides the earliest date that you can use for the new partition. To override this limitation, use the **-f** (force) parameter.

- Use the **-u** (unit) parameter to specify the time unit for the partition.
- Use the **-q** (quantity) parameter to specify the number of partitions to add. Enter an integer greater than 0.

  By default, you can add the following number of partitions:

  – No more than 60 daily partitions

  – No more than 15 weekly partitions

  – No more than 6 monthly partitions

  You can change these defaults by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".
- Use the **-c** (storable class) parameter to specify the class of objects to be stored in the partition. The default is **event**.

  > **Note:**
  >
  > To specify a storable class other than the event, bill, invoice, item, journal, newsfeed, sepa, and user activity storable classes, partitioning must have been enabled for that storable class in one of the following ways:
  >
  > – By editing the **pin_setup.values** file during the BRM installation process. See "Enabling Different Classes for Partitioning during Installation" in *BRM Installation Guide*.
  >
  > – By using the partitioning package to convert the storable class after BRM is installed. See "Converting Nonpartitioned Classes to Partitioned Classes" in *BRM System Administrator's Guide*.
- Use the **-w** (width) parameter to specify the number of units in a partition (for example, a partition that is 3 days wide, 2 weeks wide, or 1 month wide). Enter an integer greater than 0. The default is **1**.

  By default, you can use the following maximum widths:

  – 5 days

  – 3 weeks

  – 2 months

  You can change these defaults by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".
- Use the **-f** (force) parameter to create a partition with a *start date* that falls within the time period of the current partition. The existing partition is split in two:

  – One new partition contains objects created before the specified start date.

  – The other new partition contains objects created on or after the specified start date.

  Before you use this parameter:

  – For real-time partitions, stop all BRM server processes.

  – For delayed-event partitions, stop all delayed-event loading.

> **Note:**
>
> * If you use the **-f** parameter to create partitions within the time period of the current partition and you do not stop BRM processes or loading, the BRM server may write data to a partition that is in the process of being divided. That can cause a server error, such as a write operation error.
> * The **-f** parameter works differently when you remove partitions. In that case, it forces the removal of objects that do not meet the purging criteria, such as events associated with open items, by removing the partition that contains those objects.

- Use the **-p** (print) parameter to run the utility in test mode and write an SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

For more information, see "partition_utils".

The following examples show how to add partitions.

> **Note:**
>
> Adding partitions adds the requested number of partitions and an additional partition on the specified start date unless a partition labeled with that date exists.

The following command adds five real-time event partitions, starting on July 27, 2017, and one partition for events up July 27, 2017. Each partition includes events for a two-week period:

```
partition_utils -o add -t realtime -s 07272017 -u week -w 2 -q 5
```

The following command includes the **-f** parameter. This creates partitions even if they include start dates, end dates, or both start and end dates in the current partition:

```
partition_utils -o add -t realtime -s 07272017 -u week -w 2 -q 5 -f
```

The following command adds six daily event partitions starting on July 27, 2004, and one partition for events up to July 27, 2017. Because the **-w** parameter is not used, each parameter is one day long:

```
partition_utils -o add -t delayed -s 07272017 -u day -q 6
```

The following command adds 12 monthly real-time partitions for the journal storable class starting on July 27, 2017, and one partition for journal up to July 27, 2004:

```
partition_utils -c /journal -o add -t realtime -s 07272017 -u month -q 12
```

The following command creates five weekly partitions for the journal storable class starting on July 27, 2017, and one partition for journal up to July 27, 2017:

```
partition_utils -c /journal -o add -t realtime -s 07272017 -u week -q 5
```

Enabling Delayed-Event Partitioning

To add delayed-event partitions, run the **partition_utils** utility with the **enable** operation. You must enable delayed-event partitioning before you can add delayed-event partitions.

> **Note:**
>
> - After you enable delayed-event partitioning, add partitions for delayed events *before* you use the system for production. Otherwise, delayed events are stored in **partition_last**. See "About Objects Stored in partition_last and partition_last_pin".
> - You can enable delayed partitioning only for event objects. All non-event partitions are real-time partitions.

The syntax is the following:

```
partition_utils -o enable -t delayed [-c storable_class] [-p]
```

- Use the **-c** (storable class) parameter to specify the event storable class for which you want to add delayed-event partitions. Delayed-event partitions cannot be used for non-event storable classes.

  To add delayed-event partitions for all subclasses of an event, use the percent sign (**%**) as a wildcard (for example, **-c /event/session/%**).
- Use the **-p** (print) parameter to run the utility in test mode and to write an SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

  > **Note:**
  >
  > When you enable delayed-event partitioning, the **partition_utils** utility automatically runs the **update** operation to synchronize partitioning schemes across all event tables. See "Updating Partitions".

For more information, see "partition_utils".

The following command enables delayed-event partitioning for the **/event/delayed/session/telco/gsm** storable class:

```
partition_utils -o enable -t delayed -c /event/delayed/session/telco/gsm
```

You can enable partitioning for all subclasses of an event by using the percent sign (**%**) as a wildcard:

```
partition_utils -o enable -t delayed -c /event/session/%
```

> **Note:**
>
> This operation supports only delayed events. If you specify anything other than **-t delayed**, the utility returns an error.

Disabling Delayed-Event Partitioning

If you enable delayed-event partitioning for an event storable class but then decide you do not need it, you can use **sqlplus** to remove the delayed-event partitions and then to remove the **partition_last_pin** partition.

> **Note:**
>
> If you remove partitions from a delayed-event table, remove all the delayed-event partitions before removing the **partition_last_pin** partition.

For example, to disable partitioning for the EVENT_DLYD_SESSION_TELCO_GSM_T table:

1. Use the following **sqlplus** statement to ensure that no data is in **partition_last_pin**:

   ```
   SQL> select count (*) from event_dlyd_session_telco_gsm_t partition
   PARTITION_LAST_PIN
   ```

   If **partition_last_pin** contains data, add partitions to move the data.

   > **Note:**
   >
   > You must use the **-f** parameter. See "Adding Partitions".

2. Use the following **sqlplus** statement to list the delayed partitions in the table:

   ```
   SQL> select partition_name from user_tab_partitions where table_name =
   UPPER(event_dlyd_session_telco_gsm_t) and partition_name like 'P_D_%';
   ```

3. Use the following **sqlplus** statement to remove the listed partitions individually:

   ```
   SQL> alter table event_dlyd_session_telco_gsm_t drop partition partition_name
   ```

4. Use the following **sqlplus** statement to remove **partition_last_pin**:

   ```
   SQL> alter table event_dlyd_session_telco_gsm_t drop partition PARTITION_LAST_PIN
   ```

5. Use the following **sqlplus** statement to verify that the table contains no partitions:

   ```
   SQL> select count(*) from user_tab_partitions where table_name =
   UPPER(event_dlyd_session_telco_gsm_t) and partition_name like 'P_D_%';
   ```

   This should return zero rows.

Updating Partitions

The **update** operation enables you to align table partitioning schemes for all subclass tables with the current partitioning scheme for their base storable class table. Updating partitions enforces the following rules:

- From the day after tomorrow, all tables with real-time objects will have the same real-time partitioning scheme as their base table (EVENT_T for event base storable class tables, ITEM_T for item base storable class tables, and so on).
- From the day after tomorrow, all tables with delayed events will have the same delayed-event partitioning scheme as the EVENT_T table.

You must update partitions in the following cases:

- When you add custom partitioned storable classes.
- When you extend an existing storable class with an array or substruct that adds a partitioned table.
- When you add a component that includes new partitioned storable classes.

> **Note:**
>
> The **update** operation automatically runs once when you enable delayed partitions.

To align the partitioning schemes of new or changed object tables with the partitioning schemes of their base storable classes, run the **partition_utils** utility with the **update** operation.

The syntax is the following:

```
partition_utils -o update [-c storable_class] [-p]
```

- Use the **-c** (storable class) parameter to specify the class of objects to be updated. The default is **event**.
- Use the **-p** (print) parameter to run the utility in test mode and to write an SQL statement of the operation to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

For more information, see "partition_utils".

Purging Objects by Removing Partitions

The general way to purge partitioned data is to remove one or more partitions. This frees up database space.

> **Note:**
>
> - By default, partitions that contain objects associated with open items are not removed when you remove partitions. To change this default, see "Enabling Open Items to Be Purged".
> - You can use the **-f** parameter to remove partitions even if the objects are associated with open items, but this parameter must be used with care.
> - Before removing partitions that contain objects associated with open items, verify that doing so does not contradict your business practices.

> **Tip:**
>
> Before removing partitions, run the **partition_utils** utility with the **-p** parameter to write an SQL statement that shows the partitions that will be removed. See "Running the partition_utils Utility in Test Mode".

To remove partitions, run the **partitions_utils** utility with the **remove** operation. The syntax is the following:

```
partition_utils   -o remove  -s start_date  -e end_date
                  [-c storable_class] [-t realtime|delayed] [-f] [-p]
```

- Use **-s** *start_date* to specify the start of the date range for the objects to remove. The format is *MMDDYYYY*.

  > **Note:**
  >
  > The utility keeps the object's oldest partition by default. To force the removal of the oldest partition, you must set *start_date* to **01011970**.

- Use **-e** *end_date* to specify the end of the date range for the objects to remove. The format is *MMDDYYYY*.

  All partitions that are entirely within these dates are removed. See "About Purging Objects".

  By default, you can use this operation to remove only those partitions that are older than 45 days. You can change this limitation by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file. See "Customizing Partition Limitations".

- Use **-c** *storable_class* to specify the partition to remove by base storable class. The default is **event**.

  When you remove a partition, it removes partitions in all the partitioned tables for the specified base storable class and its subclasses.

> **Note:**
>
> To purge objects other than event, item, bill, invoice, journal, newsfeed, and sepa objects, you must remove their partitions by using the **-f** parameter. Operations using this parameter cannot be undone and will remove objects that are being used. Use with caution. (You can purge event, bill, invoice, item, journal, newsfeed, and sepa objects without removing their partitions. See "Purging Objects without Removing Partitions".)

- Use **-t realtime | delayed** to remove real-time or delayed-event partitions. The default is to remove both real-time and delayed-event partitions.
- Use **-f** to remove partitions even if they contain objects associated with open items. By default, partitions that contain events associated with open items are not removed. To change this default, see "Enabling Open Items to Be Purged".

> **Note:**
>
> The **-f** parameter works differently when you add partitions. In that case, it forces the splitting of partitions even when they fall within the date range of the current partition.

- Use **-p** to run the utility in test mode and write an SQL statement that shows the partitions that will be removed to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

For more information, see "partition_utils".

The following command removes the partitions with real-time events from July 20, 2021 to September 20, 2021:

```
partition_utils -o remove -s 07202021 -e 09202021 -t realtime
```

The following command removes real-time partitions for item table entries from July 20, 2021 to September 20, 2021:

```
partition_utils -o remove -s 07202021 -e 09202021 -c /item -t realtime -f
```

Purging Objects without Removing Partitions

You can purge event, bill, invoice, item, journal, newsfeed, and sepa objects without removing their partitions. The event objects must be associated with closed items. To enable this utility to purge event objects associated with open items, see "Enabling Open Items to Be Purged".

> **Note:**
>
> - Before purging objects, run the **partition_utils** utility with the **-p** parameter to write an SQL statement that shows the partitions that will be purged. See "Running the partition_utils Utility in Test Mode".
> - If the specified start and end dates do not match the partition boundaries, only objects in partitions that are completely within the date range are purged.

To purge objects without removing partitions, run the **partitions_utils** utility with the **purge** operation. The syntax is as follows:

```
partition_utils -o purge -e end_date [-t realtime|delayed] [-p]
```

- (Optional) Use the **-s** (start date) parameter to specify the start of the date range containing the objects you want to purge. The date is inclusive. The format is *MMDDYYYY*. If a start date is not specified, all objects created on or before the end date are purged.
- Use the **-e** (end date) parameter to specify the end of the date range containing the objects you want to purge. The date is inclusive. The format is *MMDDYYYY*. See "About Purging Objects".
- Use the **-t** (type) parameter to purge real-time or delayed-event partitions. The default is to purge both real-time and delayed-event partitions.
- Use the **-p** (print) parameter to run the utility in test mode and write an SQL statement that shows the partitions that will be removed to the **partition_utils.log** file without performing any action on the database. See "Running the partition_utils Utility in Test Mode".

For more information, see "partition_utils".

The following command purges real-time events before July 20, 2004:

```
partition_utils -o purge -e 07202004 -t realtime
```

Finding the Maximum POID for a Date

You can use **partition_utils -o maxpoid** to find the maximum POID in a partition for a specified date. You may need this if you have scripts to manage partitions automatically.

The syntax is the following:

```
partition_utils -o maxpoid -s date -t realtime|delayed
```

The following command finds the maximum POID for a real-time partition:

```
partition_utils -o maxpoid -s 02012005 -t realtime
```

For more information, see "partition_utils".

Customizing Partition Limitations

The **partition_utils** utility specifies default limitations to prevent the creation of an unrealistic number of partitions. You can change these limitations by editing the *BRM_home***/apps/partition_utils/partition_utils.values** file.

The default limitation values are the following:

- Partition start date: later than the day after tomorrow, earlier than 6 months from now.
- Maximum number of partitions:
  - 60 daily
  - 15 weekly
  - 6 monthly
- Maximum widths:
  - 5 days
  - 3 weeks
  - 2 months
- Removed partitions must be at least 45 days old.
- Minimum required percent of purgeable data.
- Partitions are purged only when purgeable POIDs are greater than 70% of the total purgeable POIDs in the EVENT_BAL_IMPACT_T table.

  The valid range for this setting is from 60 to 100.
- Records are deleted only if there is a specified number in a chunk. The default is 1000 records exist.

  The valid range is from 500 to 5000.
- When purging, the DELETE_IN_PLACE method is used if the number of purgeable events is greater than 5% of the total events or if more than 10,000 purgeable records.

  You can change the number of purgeable records required. The valid range is from 1000 to 20000.

Customizing the List of Events and Items Stored in partition_historic

Nonpurgeable event and items objects are stored in **partition_historic**.

If you do not need to save all the event types stored by default in **partition_historic** or if you need to save additional storable classes of events and items, use the Storable Class Editor in Developer Center to customize the list of nonpurgeable event and item storable classes.

Use the **Non-purgeable** storable class property to specify whether an event or item is purgeable.

Note:

- The **Non-purgeable** property for event and item storable classes has no effect unless your event and item tables are partitioned.
- To avoid storing event or item objects that you do not want to purge in purgeable partitions, you must customize event and item storable classes stored in **partition_historic** *before* running BRM and generating any events or items.
- If you make an event storable class nonpurgeable, make sure that all item storable classes related to the event storable class are also nonpurgeable. See "Associating Items with Nonpurgeable Events".

For more information, see "About Purging Database Objects".

45

Converting Nonpartitioned Classes to Partitioned Classes

Learn how to convert specified nonpartitioned storable classes to partitioned storable classes in the Oracle Communications Billing and Revenue Management (BRM) database. Use this procedure if you did not enable partitioning for one or more storable classes when you installed BRM.

Topics in this document:

About Converting Nonpartitioned Classes to Partitioned Classes

If you did not enable partitioning for one or more storable classes when you installed BRM, you can do so after installation.

The partitioning conversion feature splits the table of a specified storable class in the BRM database into the following partitions:

- **partition_migrate:** Holds all objects created *before* the nonpartitioned storable classes were converted to partitioned storable classes. The BRM purge utility, **partition_utils,** cannot purge objects in this partition. To purge them, you must develop your own tools based on sound Oracle database management principles.
- **partition_historic:** Holds *nonpurgeable events* created *after* the nonpartitioned storable classes were converted to partitioned storable classes. Nonpurgeable events should not be purged from the database. See "Event and Item Objects Stored in partition_historic".
- **partition_last:** A *spillover* partition that is not intended to store objects you want to purge or preserve. If you do not add purgeable partitions to your tables *before* BRM resumes generating objects, purgeable objects created after the upgrade are stored in this partition.

> **Note:**
>
> For information on how partitioning is enabled when you *install* BRM, see "Installing BRM" in *BRM Installation Guide.*

Converting Nonpartitioned Classes to Partitioned Classes

To convert nonpartitioned storable classes to partitioned storable classes, perform these tasks:

1. Increasing Disk Space for Tables and Indexes
2. Installing the Partitioning Package
3. (Optional) Reconfiguring the Parameters
4. Merging the pin_setup.values File
5. Backing Up Your BRM Database
6. Running the Partitioning Conversion Scripts
7. Adding Purgeable Partitions to Tables
8. Restarting BRM

Increasing Disk Space for Tables and Indexes

Before adding partitions to your tables, increase the disk space allocated for the tables and indexes in your BRM database.

> **Note:**
>
> Oracle recommends that you add enough space for 6 to 12 months of data.

Installing the Partitioning Package

> **Note:**
>
> If you already installed the partitioning package, features in that package cannot be reinstalled without first being uninstalled. To reinstall a feature, uninstall it, and then install it again.

To install the partitioning package:

1. Download the software to a temporary directory (*temp_dir*).

> **Note:**
>
> - 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.

2. Make sure no users are logged in to BRM.
3. Go to the directory where the Third-Party package is installed and source the **source.me** file.

> **Note:**
>
> You must source the **source.me** file to proceed with installation. Otherwise, “suitable JVM not found" and other error messages appear.

Bash shell:

```
source source.me.sh
```

C shell:

```
source source.me.csh
```

4. Go to *temp_dir* and run the following command:

```
12.0.0_PartitionUpg_platform_32_opt.bin
```

where *platform* is **solaris, linux, aix_32,** or **hpux_ia64.**

> **Note:**
>
> You can use the **-console** parameter to run the installation in command-line mode. To use the graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

5. Follow the instructions displayed during installation. The default installation directory for the partitioning package is **opt/portal/12.0**.

> **Note:**
>
> The installation program does not prompt you for the installation directory if BRM or the partitioning package is already installed on the machine and automatically installs the package in the directory in which BRM is installed (*BRM_home*).

6. Go to the directory where you installed the partitioning package and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

(Optional) Reconfiguring the Parameters

The partitioning conversion configuration file, **partition.cfg**, controls the parameters of your conversion. If necessary, change the parameters to meet your business requirements.

To reconfigure the parameters:

1. Log in as user **pin**.
2. Open the *BRM_home***/upgrade/partition/partition.cfg** file in a text editor such as **vi**.
3. Change the default parameters as necessary. For information on each parameter, see the comments in the **partition.cfg** file.
4. Save and close the file.

Merging the pin_setup.values File

Copy any customizations from your backed-up **pin_setup.values** file into the *BRM_home***/setup/pin_setup.values** file.

Backing Up Your BRM Database

Make a complete offline backup of your BRM database, and make sure the backup is completely valid and usable. For more information on performing full database backups, see your database software documentation.

In addition to the backup, use the Oracle export utility to export all BRM tables. This helps you restore individual tables if necessary.

Running the Partitioning Conversion Scripts

Converting from a nonpartitioned storable class to a partitioned storable class should take about 30 minutes. The size of your tables does not affect the speed of this conversion.

To run the partitioning conversion scripts:

1. Go to *BRM_home***/upgrade/partitioning**.
2. Run the following command:

   ```
   perl partitioning.pl
   ```

This runs a series of scripts that perform the conversion.

Check the **log** and **pinlog** files in the directory specified by the UPGRADE_LOG_DIR parameter in your **partition.cfg** file (by default, *BRM_home***/upgrade/partition/sqllog**). These log files show how long each script took to run and list any errors that occurred.

> **Note:**
>
> If errors are reported, fix them, and rerun the script.

Adding Purgeable Partitions to Tables

Before your BRM system resumes generating data, use the **partition_utils** script to add purgeable partitions to your newly partitionable tables. See "Adding Partitions".

Restarting BRM

Start all BRM processes.

> **Note:**
>
> The *object IDs* of objects generated in your newly partitioned tables will be larger than the object IDs of objects generated in your old, nonpartitioned tables. (Depending on the table, object IDs are stored in either the POID_ID0 field or the OBJ_ID0 field.)

About the Conversion Scripts and Files

Table 45-1 lists the scripts and files used to convert your nonpartitioned storable classes to partitioned storable classes. These scripts and files are in the *BRM_home***/upgrade/partition** folder.

> **Note:**
>
> To convert custom storable classes, you might need to create additional SQL scripts. To use custom scripts in the conversion, add appropriate SQL file entries to the nonpartitioning-to-partitioning conversion configuration file, **partition.cfg**. See "(Optional) Reconfiguring the Parameters".

Table 45-1 Conversion Scripts and Files

Script or File	Description
crt_pinlog.sql	SQL script that creates the **pinlog** files.

Table 45-1 (Cont.) Conversion Scripts and Files

Script or File	Description
partition.cfg	Configuration file in which you must enter details about the Oracle database configuration before you run the conversion scripts. All the conversion Perl scripts parse this file to get the database connection parameters.
partitioning.pl	Primary Perl script for the conversion process. This script calls other SQL scripts to perform the conversion.
pin_upg_common.sql	SQL script that creates the common routines needed for the conversion.
*storable_class_name*_**tables_tobe_partitioned.sql**	Custom SQL scripts that implement partitioning for the specified storable class (*storable_class_name*). See "Converting Additional Nonpartitioned Classes to Partitioned Classes".
make_indexes_partition_ready.sql **optional_partitioning.sql**	SQL scripts that implement partitioning.
upg_oracle_functions.pl	Perl script that performs many miscellaneous conversion tasks related to the BRM database.

Converting Additional Nonpartitioned Classes to Partitioned Classes

By default, the partitioning conversion feature enables you to convert the event, bill, invoice, item, and journal storable classes.

To convert additional nonpartitioned storable classes to partitioned storable classes:

1. Create a SQL script named *storable_class_name*_**tables_tobe_partitioned.sql**, where *storable_class_name* is the name of the object type whose table you want to partition.

   For samples, see the following files in the *BRM_home***/upgrade/partition/** directory, where *BRM_home* is the directory in which BRM components are installed:

   - **/event_tables_tobe_partitioned.sql**
   - **/item_tables_tobe_partitioned.sql**
   - **/bill_tables_tobe_partitioned.sql**
   - **/invoice_tables_tobe_partitioned.sql**
   - **/journal_tables_tobe_partitioned.sql**
2. Add the new SQL script to the @SQL_PARTITION_TABLES parameter in the *BRM_home***/upgrade/partition/partition.cfg** file.
3. Run the **partitioning.pl** script, which converts the specified storable class from nonpartitioned to partitioned.
4. Run the **partition_utils.pl** script, which adds purgeable partitions.

46

About Purging Data

Learn about how to purge obsolete database objects and expired account sub-balances from your Oracle Communications Billing and Revenue Management (BRM) database as well as the impact of purging major event objects on BRM applications.

Topics in this document:

- About Purging Database Objects
- About Purging BRM Event Objects
- Enabling Open Items to Be Purged
- About Purging Account Sub-Balances

Before purging data, you should be familiar with the following:

- Basic BRM concepts. See "About Billing and Revenue Management" in *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- Database administration.

About Purging Database Objects

You can purge objects that are no longer required for daily business operations from your BRM database as follows:

- If your tables are partitioned, you can use the **partition_utils** utility to purge objects. See "Partitioning Tables".
- To purge objects from *any* type of database, you can use custom purging scripts. For more information, contact Oracle support.

Purging objects enables you to do the following:

- Delete obsolete data from your database.
- Reduce storage space in your database.
- Reduce the number of objects in your database, making it easier to upgrade to later releases of BRM.

Objects Purged by Default

By default, tables for the objects listed in Table 46-1 are automatically partitioned and purged if partitioning is enabled on your BRM system and if the objects in the partition satisfy the purging criteria.

Table 46-1 Objects That Are Automatically Partitioned and Purged

Object Type	Purging Criteria
bill	• Within specified date range • Closed
event	• Within specified date range • Associated items are closed **Note:** Event objects in the **partition_historic** partition cannot be purged even if they meet all the purging criteria.
invoice	• Within specified date range
item	• Within specified date range • Closed • Not referenced by a journal object **Note:** Item objects in the **partition_historic** partition cannot be purged even if they meet all the purging criteria
journal	• Within specified date range • Fully earned by the end date of the date range **Note:** In addition, Oracle recommends that the journal object be included in a G/L report.
newsfeed	• None
sepa	• Within specified date range • Not pending (status is not 1)

About Purging BRM Event Objects

Event objects are grouped into the following categories:

- Event Objects That Have a Balance Impact
- Event Objects That Do Not Have a Balance Impact
- Event and Item Objects Stored in partition_historic

Event Objects That Have a Balance Impact

Event objects that have a balance impact are required for the following functions:

- Billing
- Accounts receivable (A/R) operations
- Tracking session charges

The balance impacts in these objects are stored in the **EVENTS_BAL_IMPACTS_T** table. These objects are typically created by usage, cycle, and purchase events.

> **Note:**
>
> Purge these event objects only *after* you finish all billing, A/R, and session event processing for the current billing cycle.

Event Objects That Do Not Have a Balance Impact

Database objects that do not have a balance impact are not needed by BRM for rating or billing. Your business, however, might need them for auditing. Auditing objects and unused objects are in this category.

If you purge objects that do not generate a balance impact, you cannot view auditing information for those objects. For example, if you purge **/event/customer/login** objects and then want to check when a user logged in, you will not be able to find the user's login event.

> **Note:**
>
> Before purging objects that do not have a balance impact, verify that the objects are not required by any custom code.

Event and Item Objects Stored in partition_historic

If your event and item tables are partitioned, event and item objects that are required for the product to behave correctly or for auditing purposes (nonpurgeable events and items) are stored in the **partition_historic** partition.

> **Note:**
>
> - By default, all item objects are purgeable.
> - If an event is nonpurgeable, its associated items should also be nonpurgeable because the event is incomplete without them.
> - If you make an item storable class nonpurgeable, make sure the item storable class applies only to nonpurgeable events.
> - If you must purge event objects in your implementation, before purging them, make sure you understand the impact of purging those objects. See "Impact of Purging Event Objects".
>
> See "Associating Items with Nonpurgeable Events".

By default, the following types of event objects are nonpurgeable and thus are stored in **partition_historic** when you install BRM:

- **/event/billing/cdc_update**
- **/event/billing/cdcd_update**
- **/event/billing/cycle/discount**
- **/event/billing/cycle/discount/mostcalled**
- **/event/billing/cycle/rollover**
- **/event/billing/cycle/rollover/monthly**
- **/event/billing/cycle/rollover_transfer**

- **/event/billing/deal**
- **/event/billing/deal/purchase**
- **/event/billing/deal/cancel**
- **/event/billing/discount/action**
- **/event/billing/discount/action/cancel**
- **/event/billing/discount/action/modify**
- **/event/billing/discount/action/modify/status**
- **/event/billing/discount/action/purchase**
- **/event/billing/discount/action/set_validity**
- **/event/billing/lcupdate**
- **/event/billing/mfuc_update**
- **/event/billing/ordered_balgrp**
- **/event/billing/product/action**
- **/event/billing/product/action/cancel**
- **/event/billing/product/action/modify**
- **/event/billing/product/action/modify/status**
- **/event/billing/product/action/purchase**
- **/event/billing/product/action/set_validity**
- **/event/delayed/tmp_profile_event_ordering**
- **/event/group/sharing**
- **/event/group/sharing/charges**
- **/event/group/sharing/discounts**
- **/event/group/sharing/profiles**

You can customize this list to meet your business requirements. For more information on **partition_historic**, see "About Partitioning".

Impact of Purging Event Objects

Before purging event objects, you must know this information:

- Impact of purging the event objects
- How long to keep the event objects

The following tables describe the impact of purging event objects that contain data for billing, A/R, opening and closing sessions, and auditing.

> **Note:**
>
> Only major event storable classes and events stored by default in the **partition_ historic** tables are listed in this document.

Billing Event Objects

Table 46-2 lists the impacted Billing Event Objects.

Table 46-2 Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/cdc_update	If you purge these events, the number of contract days for resources will be lost. You cannot apply discounts based on the cumulative number of contract days for resources.
/event/billing/cdcd_update	If you purge these events, the number of contract days for discount resources will be lost. You cannot apply discounts based on the cumulative number of contract days for discount resources.
/event/billing/charge and all of its subclasses	If you purge these event objects: • You cannot use the **resubmit** parameter with the **pin_recover** utility to resubmit failed credit card transactions. • The **pin_clean** utility cannot report VERIFY checkpoints. If you have run the **pin_recover** and **pin_clean** utilities and successfully recovered all failed credit card transactions, you can purge these event objects.
/event/billing/cycle and all of its subclasses	If you purge these event objects: • Invoices cannot display details about subscription charges applied to accounts. • Data in default reports generated by the **pin_ledger_report** utility will not tally correctly because the reports display charges accrued by these objects. • You cannot perform A/R activities such as adjustments, write-offs, disputes, and settlements for subscription charges logged by these events. If you generate invoices and G/L reports, keep these event objects through the current billing cycle.
/event/billing/cycle/discount/mostcalled	If you purge these events, all the information about the total charges, duration, and the number of calls for the most called numbers will be lost. You cannot apply the most-called number discounts. See also the impact of purging **/event/billing/cycle** and all of its subclasses.
/event/billing/cycle/rollover/monthly	Purging these objects will have the following impact: • Rollover correction during final billing will not work if the rollover events for the current cycle have been purged. • Rerating will not work correctly if the rollover events in the rerating period have been purged. • The controlled rollover of free resources during plan transition will not work if the rollover events for the current cycle have been purged. See also the impact of purging **/event/billing/cycle** and all of its subclasses.
/event/billing/cycle/rollover_transfer	If you purge these events, information about the balance impacts of the original rollover event will be lost. See also the impact of purging **/event/billing/cycle** and all of its subclasses. You may want to keep these objects for auditing purposes.

Table 46-2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/deal	Abstract class, not persisted in the database.
/event/billing/deal/cancel	If you purge these events, rerating, which needs to replay these events, will not work correctly. However, you can purge old events that occurred before your rerating time frame.
/event/billing/deal/purchase	If you purge these events, rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame. You cannot see deal purchase history. Keep these event objects as long as you want to display the history of deal purchases.
/event/billing/debit	If you purge these event objects: • Invoices cannot display event details for debit or credit items completed during the current billing cycle. • If you generate any custom reports that use debit events, the reports will be incorrect. • If you configure the **pin_ledger_report** utility to report debits and credits, the reported numbers will be incorrect. For information about the **pin_ledger_report** utility, see "pin_ledger_report" in *BRM Collecting General Ledger Data*. Keep these event objects as long as you want to do the following: • Display debit events. • Display event details in an invoice. • Create custom reports that use debit events.
/event/billing/discount/action/cancel	The PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS and PCM_OP_ SUBSCRIPTION _GET_HISTORY opcodes refer to these events to show the history of the discount. If these event objects are purged, you cannot see the corresponding discount history. You can, however, see the discount history contained in other types of event objects that are not purged. If these are the only events for the discount, then no history will be shown
/event/billing/discount/action/modify	The PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS and PCM_OP_ SUBSCRIPTION _GET_HISTORY opcodes refer to these events to show the history of the discount. If these event objects are purged, you cannot see the corresponding discount history. You can, however, see discount history contained in other types of event objects that were not purged. Keep these event objects if you need to display the history of discount attribute and status modification.
/event/billing/discount/action/modify/ status	If these event objects are purged, you cannot see the corresponding discount history. You can, however, see discount history contained in other types of event objects that were not purged. Keep these event objects if you need to display the history of discount attribute and status modification.
/event/billing/discount/action/purchase	If you purge these event objects, you cannot see the corresponding discount history. Keep these event objects as long as you want to display the history of discount purchases.

Table 46-2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/discount/action/set_validity	If you purge these events: • Rating will not consider the validity period setting because the discount validity period settings prior to first usage and the new calculated validity period after first usage will be lost. • Rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame.
/event/billing/lcupdate	If this event is purged, discount amount based on the number of active subscriptions will not be applied.
/event/billing/mfuc_update	If this event is purged, monthly fee and usage discount will not be applied at billing time. You may also want to keep these objects for auditing purposes.
/event/billing/ordered_balgrp	Abstract class, not persisted in the database.
/event/billing/ordered_balgrp/create	If you purge these events, the history of the creation of the object is lost. You may also want to keep these objects for auditing purposes.
/event/billing/ordered_balgrp/delete	If you purge these events, the history of the deletion of the object is lost. You may also want to keep these objects for auditing purposes.
/event/billing/ordered_balgrp/modify	If you purge these events, the history of the modifications of the object is lost. You may also want to keep these objects for auditing purposes.
/event/billing/product	If you purge these event objects: • Reports generated by the **pin_ledger_report** utility cannot display purchase and cancellation fees, and data shown in these reports will be incorrect. • Invoices cannot display purchase or cancellation fees. Keep these event objects if either of the following is true: • You use Billing Care to view current or historical information for canceled and item products. • Your custom applications use these event objects.
/event/billing/product/action/cancel	If you purge these events, rerating, which needs to replay these events, will not work correctly. However, you can purge old events that occurred before your rerating time frame.
/event/billing/product/action/modify **/event/billing/product/action/modify/status**	If you purge these events: • Rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame. • You cannot see the corresponding product history. You can, however, see product history contained in other types of product event objects that were not purged. Keep these event objects if you need to display the history of product attribute and status modification.
/event/billing/product/action/purchase	If you purge these events: • Rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame. • You cannot see the corresponding product history. Keep these event objects as long as you want to display the history of product purchases.

Table 46-2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/product/action/set_validity	If you purge these events: • Rating will not consider the validity period setting because the product's validity period settings prior to first usage and the new calculated validity period after first usage will be lost. Also during the back-out process, rerating uses these events to remove the validity setting. **Note:** You can purge old events that occurred before your rerating time frame.
/event/billing/product/fee/cycle/ cycle_forward_ annual **/event/billing/product/fee/cycle/ cycle_forward_ bimonthly** **/event/billing/product/fee/cycle/ cycle_forward_ monthly**	If you purge the cycle_forward_* events for a particular period, features or operations that involve cancellation will not work correctly. For example, if all the cycle_forward_* events from 01/01/2008 to 06/01/2008 are purged, then operations, such as product or discount cancellations, service inactivations, plan or deal transitions, and change of options, which internally call the cancellations of the product or discount during that time period will not give correct results. During product and discount cancellations, BRM searches for all the original charged events and refunds the charges for each event individually. If these events are purged, then charges for these events will not be refunded.
/event/billing/validate/cc	You may want to keep these objects for auditing purposes, for example, to keep track of credit card validations.

Accounts Receivable Event Objects

Table 46-3 lists the Accounts Receivable Event Objects.

Table 46-3 Accounts Receivable Event Objects

Event object	Impact of purging/when to purge
/event/billing/adjustment/account **/event/billing/adjustment/item**	If you purge these event objects, you lose some G/L information. The **pin_ledger_report** utility uses these event objects to generate the G/L report for **billed_earned** and **billed** revenue types. For information about the **pin_ledger_report** utility, see "pin_ledger_report" in *BRM Collecting General Ledger Data*. Keep these event objects for the specified G/L posting period.
/event/billing/adjustment/event	If you purge these event objects, you lose some G/L information. The **pin_ledger_report** utility uses these event objects to generate the G/L report for **billed_earned** and **billed** revenue types. Keep these event objects for the specified G/L posting period.
/event/billing/adjustment/tax_event	If you purge these event objects, you lose some G/L information. The **pin_ledger_report** utility uses these event objects to generate the G/L report for **billed_earned** and **billed** revenue types. Keep these event objects for the specified G/L posting period.

Table 46-3 (Cont.) Accounts Receivable Event Objects

Event object	Impact of purging/when to purge
/event/billing/batch/payment	If you purge these event objects: • You cannot use the **pin_recover** utility with the **resubmit** parameter to resend failed transactions. • You cannot use the **pin_clean** utility. You can purge the original batch payment event objects after you successfully resubmit any failed batch transactions.
/event/billing/batch/reversal	You can purge these event objects if you do not have any custom applications using them.
/event/billing/dispute/item **/event/billing/settlement/item**	If you purge these event objects, you lose some past G/L, dispute, and settlement information. The **pin_ledger_report** uses these event objects to generate the G/L report for **billed_earned** and **billed** revenue types. For information about the **pin_ledger_report** utility, see "pin_ledger_report" in *BRM Collecting General Ledger Data*. Keep these event objects for the specified G/L posting period.
/event/billing/item/transfer	If you purge these event objects: • You cannot audit the related bill items. • A/R Tool might not show the actions performed on the related bill items. • Payments cannot be reversed. You must preserve these event objects if you intend to reverse payments at a future date. To find the items a payment was applied to, the PCM_OP_BILL_REVERSE_PAYMENT opcode searches for all the transfer events in which PIN_FLD_ITEM_OBJECT is equal to the payment item's POID. Keep transfer event objects as long as you keep their related billable items.
/event/billing/payment and all of its subclasses	If you purge these event objects: • You cannot perform payment reversals. • If you generate G/L reports with the **pin_ledger_report** utility, the reports do not show that payments recorded in the purged events have been made. By default, G/L reports show payment data. • Invoices for the current billing cycle cannot show event details for payments. • Payment Tool cannot display payment information. • Custom reports that use payment objects will be incorrect. If you have already reversed payments or your company is bound by corporate policies prohibiting payment reversal after a specified period of time, you can purge these events.

Table 46-3 (Cont.) Accounts Receivable Event Objects

Event object	Impact of purging/when to purge
/event/billing/reversal and all of its subclasses	If you purge these event objects: • If the **pin_ledger_report** utility is configured to report reversals, G/L reports will be incorrect for periods whose events were purged. For information about the **pin_ledger_report** utility, see "pin_ledger_report" in *BRM Collecting General Ledger Data*. • Invoices for the current billing cycle cannot display event details for the purged payment reversals. • Any custom reports that use the purged reversal objects will be incorrect. They cannot display details of event reversals. Your business requirements determine how long you keep these event objects.
/event/billing/writeoff/account **/event/billing/writeoff/bill** **/event/billing/writeoff/item** **/event/billing/writeoff/tax_account** **/event/billing/writeoff/tax_bill** **/event/billing/writeoff/tax_item**	If you purge these event objects, you lose some G/L and tax information. The **pin_ledger_report** utility uses these event objects to generate the G/L report for **billed_earned** and **billed** revenue types. Keep these event objects for the specified G/L posting period.

Delayed Event Objects

Table 46-4 lists the Delayed Event Objects.

Table 46-4 Delayed Event Objects

Event object	Impact of purging/when to purge
event/delayed/ tmp_profile_event_ordering	If you purge these objects, out-of-order events will not be loaded into the **/profile/event_ordering** object; therefore, out-of-order events will not be detected or processed.

Group Event Objects

Table 46-5 lists the Group Event Objects.

Table 46-5 Group Event Objects

Event object	Impact of purging/when to purge
/event/group/member	Purge these event objects if you do not need to track the addition or deletion of member accounts.
/event/group/parent	Purge these event objects if you do not need to track the creation of parent accounts.

Sharing Group Event Objects

Table 46-6 lists the Sharing Group Event Objects.

Table 46-6 Sharing Group Event Objects

Event object	Impact of purging/when to purge
/event/group/sharing/charges/create **/event/group/sharing/charges/delete** **/event/group/sharing/charges/modify**	Purging these objects does not affect the behavior of the product. You may want to keep these objects for auditing purposes, because if you purge these events, the history of creation, deletion, and modification of the **/group/sharing/charges** object is lost.
/event/group/sharing/discounts/create **/event/group/sharing/discounts/delete** **/event/group/sharing/discounts/modify**	Purging these objects does not affect the behavior of the product. You may want to keep these objects for auditing purposes, because if you purge these events, the history of creation, deletion, and modification of the **/group/sharing/charges** object is lost.

Session Event Objects

Table 46-7 lists the Session Event Objects.

Table 46-7 Session Event Objects

Event object	Impact of purging/when to purge
/event/session and all of its subclasses	If you purge these event objects: • And if you do not purge the corresponding item objects, the item objects will be inaccurate. • Different event objects point to session event objects (**session_obj_*** in **EVENT_T**) for audit purposes. You lose auditing information if you purge session event objects. • You cannot make adjustments or settle disputes on bills containing the corresponding items. • You cannot use invoicing, taxation, bill adjustments, or G/L functionality. • If you purge **/event/session/dialup** objects, batch accounting operations might be performed more than once, which could result in customers being overcharged. **Note:** For dialup events, the CM **pin.conf** file contains an entry called **trans_id_window**. Purging event objects older than **now - trans_id_window** does not affect whether batch accounting occurs more than once. Most session event objects have balance impacts and are required for billing, A/R, and opening and closing session activity. Therefore, these objects should be kept for at least one accounting cycle.

Auditing Event Objects

Table 46-8 lists the Auditing Event Objects.

Table 46-8 Auditing Event Objects

Event object	Impact of purging/when to purge
/event/audit/price **/event/audit/price/deal** **/event/audit/price/discount** **/event/audit/price/plan** **/event/audit/price/product** **/event/audit/price/rate** **/event/audit/price/rate_plan** **/event/audit/price/rate_plan_selector**	Purging these objects does not affect the behavior of the product. These event objects are generated by BRM auditing features. Keep these event objects as long as you need the auditing information they contain.

Enabling Open Items to Be Purged

BRM purges item objects and their associated event objects only when the item objects are closed. In some accounts, such as prepaid, items are rarely closed. Hence, events associated with the open items also cannot be purged. The unpurgeable items and events take up storage space. To free up storage space, you can make such items purgeable and then purge them.

To enable open item objects to be purged:

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

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

2. In the business profile associated with bill units whose open items you want to close, add the following line:

   ```
   <NameValue key="AutoClose_Billed_Items" value="Yes" />
   ```

   The status of all open items associated with the bill units in *future* billing cycles will now be set to closed, and their due will be set to 0 at the end of each billing cycle.

   To close open item objects processed in *past* billing cycles, see "Closing Open Item Objects Processed in Past Billing Cycles".

3. Save and close the file.

4. Run the following command, which loads the updated contents of the **pin_business_profile.xml** file into the BRM database:

   ```
   load_pin_business_profile -v pin_business_profile.xml
   ```

5. Read the object with the **testnap** utility or Object Browser and verify your changes.

   For general instructions on using **testnap**, see "testnap" in *BRM Developer's Guide*.

6. Stop and restart the CM.

For more information about business profiles, see "Creating and Managing Business Profiles" in *BRM Managing Customers*.

Closing Open Item Objects Processed in Past Billing Cycles

Setting the **AutoClose_Billed_Items** key-value pair in a bill unit's business profile to **Yes** closes all items associated with the bill unit in *future* billing cycles at the end of each cycle.

To close open item objects processed in *past* billing cycles:

1. Verify that the **AutoClose_Billed_Items** key-value pair in the business profile associated with the bill units whose open items you want to close is set to **Yes**:

   ```
   <NameValue key="AutoClose_Billed_Items" value="Yes" />
   ```

   If that line is not in the business profile, add it. See "Enabling Open Items to Be Purged".
2. Open the *BRM_home*/**apps/partition_utils/partition_utils.values** file in a text editor.
3. Do the following:
   - Set the COMMIT_BUNDLE_SIZE parameter to the number of items closed before the changes are committed to the database.

     Higher numbers reduce the frequency of commit calls but increase the time that table rows are locked. Oracle recommends using a high number (such as the default 10000) if the number of items to close is high (for example, billions of items must be closed).
   - Configure the database connection parameters. See "Configuring a Database Connection".
4. Save and close the file.
5. In the *BRM_home*/**apps/partition_utils** directory, run the following command, which closes all open items of the bill units whose business profile contains the **AutoClose_Billed_Items** tag set to **Yes**:

   ```
   pin_close_items
   ```

For more information, see "partition_utils".

About Purging Account Sub-Balances

You can purge expired account sub-balances that are no longer required for daily business operations from your BRM database. Expired sub-balances are the validity-based balances whose valid end dates are in the past.

> **Note:**
>
> When you delete sub-balances from the database, events that impacted those sub-balances cannot be rerated. Make sure you no longer need the expired sub-balances before deleting them.

You purge expired sub-balances by running the **pin_sub_balance_cleanup** utility. Run this utility from the *BRM_home*/**apps/pin_subscription** directory, which contains the appropriate configuration file.

Run the utility with one of the following commands:

- To purge sub-balances that expired more than a specified number of days prior to today, run this command. For example, enter 60 to purge sub-balances that are older than 60 days.

  pin_sub_balance_cleanup -n *number_of_days* [**-t**]

- To purge sub-balances that expired prior to a specified date, run this command. Use the format *MM*/*DD*/*YYYY*. For example, enter 06/30/2023 to delete expired sub-balances older than June 30, 2023.

  pin_sub_balance_cleanup -d *date* [**-t**]

> **Note:**
>
> Include the **-t** parameter to purge sub-balances with temporary credit limits.

For more information, see "pin_sub_balance_cleanup".

47

Generating Virtual Columns on Event Tables

Learn how to generate virtual columns in the Oracle Communications Billing and Revenue Management (BRM) database. In the current release, virtual columns are generated only for event tables (for the **/event** storable class and subclasses).

Topics in this document:

About Generating Virtual Columns on Event Tables

Oracle Database 12*c* allows you to create *virtual columns* on tables. A virtual column is similar to a normal table column but it is defined by an expression. The result of evaluation of this expression becomes the value of the column. A virtual column contains a function upon other table columns. Virtual columns are not physically stored in the table (they are derived from data in the other columns of the table) and their values are computed at run time when you query the data. Being able to create virtual columns is enabled by default in Oracle Database 12*c*. See the Oracle Database documentation for detailed information about virtual columns.

Implementations of BRM have shown that a high percentage of the BRM database storage space can be used by the event tables. BRM can use virtual columns in a way that results in space savings for event records. To use virtual columns in the BRM database, you convert event storable classes (**/event** and its subclasses) in the BRM schema to use virtual columns. You convert event storable classes to use virtual columns by running the **pin_virtual_gen** utility (see "Generating Virtual Columns on Event Tables" for instructions). The savings in database storage applies to event data that the system creates *after* the virtual columns are generated (not to existing event data). Virtual column functionality is transparent to the BRM application.

BRM creates virtual columns on the POID *field_name*_**type** columns of event tables in the BRM database. The POID *field_name*_**type** columns are the columns that are SQL mappings for the PIN_FLDT_POID type (the POID data type). After the virtual columns are enabled, a new column is added named *field_name*_**type_id** that stores the ID mapping for the value in the POID *field_name*_**type** column.

For example, these fields within the event storable classes are candidates for conversion to virtual columns:

- **PIN_FLD_POID**

  The **poid_type** column is converted to a virtual column.

- **PIN_FLD_ACCOUNT_OBJ**

  The **account_obj_type** column is converted to a virtual column.

After the event storable classes have been enabled to have virtual columns, any new event subclass (for example, **/event/billing/discount/new**) will be virtual-column enabled. Specifically, all PIN_FLDT_POID *field_name*_**type** columns within the new event subclass will use virtual columns. Virtual columns cannot be enabled only for a subclass (the base storable class must be virtual-column enabled).

> **Note:**
>
> If you have a nonpartitioned schema, and you want to enable partitions, you must enable the partitions *before* enabling virtual columns. After you enable virtual columns on a nonpartitioned schema, you cannot enable partitioning.

See "Generating Virtual Columns on Event Tables" for instructions on converting event storable classes in the BRM schema to use virtual columns.

For additional information on how using virtual columns in your BRM database may impact your system, see the following documentation:

- See "About Adding Virtual Column Support to Your Applications" in *BRM Developer's Guide*.
- See "Creating Control Files for Custom Event Tables in a Virtual Column-Enabled System" in *BRM Migrating Accounts to the BRM Database*.

Generating Virtual Columns on Event Tables

This section describes how to generate virtual columns in the BRM database for event tables (for the **/event** storable class and subclasses).

You use the **pin_virtual_gen** utility to convert a standard BRM database into one with virtual columns. The utility generates virtual columns on event tables for all event storable classes.

For information about using virtual columns in BRM, see "About Generating Virtual Columns on Event Tables".

For information about **pin_virtual_gen** syntax, see "pin_virtual_gen".

Before you can generate virtual columns, do the following:

- Plan for downtime of your BRM system.

  The duration of time for running the utility that generates virtual columns is a downtime for the BRM system.

  > **Note:**
  >
  > If you have a nonpartitioned schema, and you want to enable partitions, you must enable the partitions *before* enabling virtual columns. After you enable virtual columns on a nonpartitioned schema, you cannot enable partitioning. See "Converting Nonpartitioned Classes to Partitioned Classes" for more information.

- (Optional) The **pin_virtual_gen** utility uses an **Infranet.properties** file which is preconfigured with the required values to run it. For information on setting the log level and number of threads for the utility, see "pin_virtual_gen".

To generate virtual columns on event tables:

1. Go to *BRM_home*/**apps/pin_virtual_columns**.
2. While your BRM system is processing, run the following command.

   ```
   pin_virtual_gen -gentasks verify_types -execute
   ```

   The utility checks if you have POID custom type names for your custom storable classes.

   This can be a long-running process.
3. Do one of the following:
   - If the utility returns a message that it found *no invalid object names*, stop the BRM system.
   - If the utility returns a message that it found object names *that are missing from the data dictionary*, do the following:
     a. Stop the BRM system.
     b. Run the following command:

        ```
        pin_virtual_gen -gentasks create_types -execute
        ```

        This command stores the POID custom type names of custom storable class types in the data dictionary of the BRM schema (required for a virtual column-enabled system).
4. Run the following command:

   ```
   pin_virtual_gen -gentasks create -execute
   ```

   This command converts the tables within the **/event** storable class and its subclasses to use virtual columns.

   If the **pin_virtual_gen** utility is interrupted while running this command, you can run the following command:

   ```
   pin_virtual_gen -readtasks create -execute
   ```
5. Start the BRM system.

For more information about using the **pin_virtual_gen** utility, see "pin_virtual_gen" and "Viewing Tasks for Generating Virtual Columns".

Viewing Tasks for Generating Virtual Columns

When you run the **pin_virtual_gen** utility, various tasks (jobs) are run to generate the virtual columns. The statuses of the tasks are maintained in the database so if the utility is interrupted, you can restart it without any issue.

Each task has a task ID. You can view the tasks before or after they are run (before or after they have generated virtual columns). You can view task details of all tasks or only tasks within a task ID range.

You may want to view tasks at the following times:

- Before you generate virtual columns, when you want to see what is going to happen to your database. By viewing the SQL statements of the tasks, you can see which tables will have virtual columns added to them and which tables will be renamed.

  See "Viewing and then Running Virtual-Column Tasks".

- After you generate virtual columns, when you want to see the tasks that completed. For example, if the **pin_virtual_gen** utility is interrupted while running, you might want to view the tasks that ran before the interruption.

  See "Viewing Virtual-Column Task Details".

For more information, see "pin_virtual_gen".

Viewing and then Running Virtual-Column Tasks

To view virtual-column tasks and then run them:

1. Run the following command:

   ```
   pin_virtual_gen -gentasks create
   ```

   This command generates the tasks and stores them in the database without executing them.

2. Do one of the following:

   - To view task details of all tasks, run the following command:

     ```
     pin_virtual_gen -showtasks
     ```

   - To view task details for tasks within a task ID range, run the following command:

     ```
     pin_virtual_gen -showtasks [minID maxID]
     ```

     where you want to see tasks that have an ID greater than *minID* and less than *maxID*.

3. After viewing the tasks, run them by entering the following command:

   ```
   pin_virtual_gen -readtasks create -execute
   ```

   This command reads the tasks from the database and runs them.

> **Note:**
>
> If there is an error before the job completes after running either the **gentasks** or the **readtasks** command (for example, if there is a power outage), run the following command:
>
> ```
> pin_virtual_gen -readtasks create -execute
> ```
>
> The **readtasks** command reads the statuses of the various tasks recorded in the database and runs the appropriate tasks.

Viewing Virtual-Column Task Details

To view the details of all virtual-column tasks, run the following command:

```
pin_virtual_gen -showtasks 0 -
```

To view the details of virtual-column tasks by task ID, run the command:

```
pin_virtual_gen -showtasks [minID maxID]
```

where you want to see tasks that have an ID greater than *minID* and less than *maxID*.

If you omit *minID* and *maxID*, the details of all tasks are displayed.

Setting Up Virtual Columns for RE Loader

RE Loader populates event tables. After you generate virtual columns on event tables in your BRM installation, you must run the **pin_gen_classid_values.pl** script. Running the script ensures that the proper mapping of BRM object types and their corresponding object IDs is created for your extended event objects in a virtual column-enabled system.

To set up RE Loader for virtual column-enabled systems:

1. Go to *BRM_home***/setup/scripts**.
2. Open the **pin_gen_classid_values.pl** file and verify that the first line in the file is pointing to the location of Perl in your installation.
3. Run the Perl script **pin_gen_classid_values.pl**.

   Running the script regenerates the **classid_values.txt** file that is used by RE Loader. The **classid_values.txt** file has the mapping of BRM object types (**poid_types**) and their corresponding object IDs (**object_ids**).

If you have extended BRM objects and these extended objects are new event subclasses that impact RE Loader, you must create new SQL Loader (**sqlldr**) control files. In virtual column-enabled systems, the RE Loader **sqlldr** control files must use the keywords VIRTUAL_CHAR and VIRTUAL_CONSTANT in the section that specifies the data definition of rows and also in the constant section.

Exporting a BRM Schema with Virtual Columns

After you generate virtual columns, if you need to export your BRM schema, you must remove the virtual columns from the schema before the export. In addition, if you need to restore the exported schema, you need to add the virtual columns back after the import.

> **Note:**
>
> Ensure that you use the Oracle Database export and import utilities that support virtual columns. Refer to the Oracle Database 12*c* documentation for information.

To export a BRM schema with virtual columns:

1. Stop the BRM system.

2. Go to *BRM_home*/**apps/pin_virtual_columns** and run the following command:

   ```
   pin_virtual_gen -gentasks pre_export -execute
   ```

   This command removes the virtual columns.

3. Export the schema using the Oracle Database export utility (for example, run the **expdp** command).

   The BRM schema is now exported into a dump file and has no virtual columns.

4. Go to *BRM_home*/**apps/pin_virtual_columns** and run the following command:

   ```
   pin_virtual_gen -gentasks post_export -execute
   ```

   This command restores the virtual columns (the BRM schema virtual columns are again enabled).

5. Start the BRM system.

6. If you need to restore the database by importing the schema from the dump file back to disk, do the following:

   a. Stop the BRM system.

   b. Import the schema using the Oracle Database import utility (for example, run the **impdp** command).

      The imported schema does not have the virtual columns because you removed them when you exported the schema to the dump file.

   c. Go to *BRM_home*/**apps/pin_virtual_columns** and run the following command:

      ```
      pin_virtual_gen -gentasks post_export -execute
      ```

      This command restores the virtual columns in your restored database (the BRM schema virtual columns are again enabled).

   d. Start the BRM system.

Part VII

Managing a Multischema System

This part describes how to manage an Oracle Communications Billing and Revenue Management (BRM) multischema system. It contains the following chapters:

- Managing a Multischema System
- Multischema Utilities

48

Managing a Multischema System

Learn how to manage an Oracle Communications Billing and Revenue Management (BRM) multischema system.

Topics in this document:

Before you read this document, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

About Multischema Systems

Multidatabase Manager is an optional feature that supports multiple database schemas in a single BRM installation. A multischema system lets you distribute your customer accounts among several schemas, providing increased storage capacity. For an overview, see "A BRM Multischema Production System" in *BRM Installation Guide*.

You can run billing and invoicing utilities against each schema. For more information, see "Setting Up Billing to Run in a Multischema Environment" in *BRM Configuring and Running Billing*.

Converting a Single-Schema System to a Multischema System

To convert a single-schema system to a multischema system, see "Installing a Multischema System" in *BRM Installation Guide*.

> **Note:**
>
> For multischema systems, Oracle recommends installing the BRM database in an Oracle Real Application Clusters (Oracle RAC) system.

Preparing to Manage a Multischema System

Before working with your multischema system, verify that the **PERL5LIB** environment variable is a system variable and contains *BRM_home***/perl/lib/5.8.0**.

Adding a BRM Installation Machine to a Multischema Environment

If you already have a multischema system configured and you want to configure an additional BRM installation machine to run a CM in a multischema environment:

1. Download and install the Third-Party software package on the new machine.
2. Download and install the BRM software on the new machine.

   See "Installing BRM on a Secondary Installation Machine" in *BRM Installation Guide*.
3. Download and install Multidatabase Manager on the new machine.

   > **Caution:**
   >
   > *Do not* run the **pin_multidb.pl** script, which reconfigures your entire multischema system.

   See "Installing and Configuring Multidatabase Manager on the Primary Installation Machine" in *BRM Installation Guide*.
4. Configure the BRM software by editing the **pin_setup.values** file.

   > **Note:**
   >
   > In the **pin_setup.values** file, make sure that the following database parameters are set up correctly:
   >
   > ```
   > $SETUP_DROP_ALL_TABLES = "No";
   > $SETUP_INIT_DB = "No";
   > ```
5. Run the **pin_setup** script.

Adding Database Schemas to a Multischema System

To add one or more database schemas to a multischema system:

1. Add a schema to your Oracle RAC system.

   For more information, see the following:

   - “Setting Up Oracle RAC for a Multischema System" in *BRM Installation Guide*
   - Oracle RAC documentation
2. Download and install the Third-Party software package on a new secondary installation machine.
3. Install BRM on the new secondary installation machine.

   See “Installing BRM on a Secondary Installation Machine" in *BRM Installation Guide*.
4. Verify that the new secondary installation machine can connect to the new database schema.

   See "Verifying Access between Secondary Installation Machine and Secondary Schemas" in "Using testnap to Verify Access to Your Schemas" in *BRM Installation Guide*.
5. Configure the **pin_multidb.conf** file on the primary installation machine for the new schema.

   See "Configuring the pin_multidb.conf File on the Primary Installation Machine".
6. Run the **pin_multidb.pl** script on the *primary* installation machine.

   See "Running pin_multidb.pl on the Primary Installation Machine" in *BRM Installation Guide*.

> **Note:**
>
> During this procedure, when you run **pin_multidb.pl -i**, you are prompted to configure the Oracle DM on the new secondary installation machine to use schema qualifications. For information, see "Setting Up Schema Qualifications" in *BRM Installation Guide*.

Configuring the pin_multidb.conf File on the Primary Installation Machine

To configure the new secondary database schema, perform the following procedure on the primary installation machine:

1. Open the *BRM_home***/setup/scripts/pin_multidb.conf** file in a text editor.
2. Modify the following parameters to indicate which database schema instances you are adding to the system.

   - $PIN_MD_SECONDARY_START_INST
   - $PIN_MD_SECONDARY_END_INST

   For example, if your system contains three secondary database schemas (0, 1, and 2) and you are adding two schemas, set **$PIN_MD_SECONDARY_START_INST** to **3**

and **$PIN_MD_SECONDARY_END_INST** to **4**. That tells the **pin_multidb.pl** script to configure secondary database schemas based only on **$PIN_MD_SECONDARY*[*x*]** arrays 3 and 4 and to ignore arrays 0, 1, and 2:

```
$PIN_MD_SECONDARY_START_INST    = "3";
$PIN_MD_SECONDARY_END_INST      = "4";
```

3. Modify the following entries for each new schema, making sure you do the following:
   - Create a set of **$PIN_MD_SECONDARY*[*x*]** entries for each schema you add to the system.
   - Update the array number (*x*) to the appropriate value.
   - Update the database number for the secondary database schema.
   - Update the secondary database schema alias.
   - Update the host name of the secondary DM machine.
   - Update the DM port number for the secondary database schema.

```
                                #-------------------------
                                # Secondary db information
                                #-------------------------
$PIN_MD_SECONDARY_DBNO     [0] = "0.0.0.3";
$PIN_MD_SECONDARY_OWNER    [0] = "pin";
$PIN_MD_SECONDARY_PASSWD   [0] = "pin";
$PIN_MD_SECONDARY_DBNAME   [0] = "pindbhostname";
$PIN_MD_SECONDARY_HOSTNAME [0] = "server3";
$PIN_MD_SECONDARY_PORT     [0] = "11952";       # dm port
                                #-------------------------
                                # Secondary db information
                                #-------------------------
$PIN_MD_SECONDARY_DBNO     [1] = "0.0.0.4";
$PIN_MD_SECONDARY_OWNER    [1] = "pin";
$PIN_MD_SECONDARY_PASSWD   [1] = "pin";
$PIN_MD_SECONDARY_DBNAME   [1] = "pindbhostname";
$PIN_MD_SECONDARY_HOSTNAME [1] = "server4";
$PIN_MD_SECONDARY_PORT     [1] = "11953";       # dm port
```

4. Save and close the file.

Setting Database Schema Status

Database schema status determines whether a schema is available for account creation. Schemas can be set to open, closed, or unavailable:

- **Open:** Open schemas are available for account creation.
- **Closed:** Closed schemas are not used for account creation under most circumstances. Accounts are created in a closed schema only if a sponsoring account belongs to that schema or if all schemas are closed. If all schemas are closed, Multidatabase Manager chooses a closed schema at random in which to create accounts and continues to create accounts in that schema until a schema becomes open. To limit the number of accounts created in a schema, you can manually change the schema's status to closed, or you can have Multidatabase Manager automatically change it to closed when the schema reaches a predefined limit.

- **Unavailable:** Unavailable schemas are not used for account creation unless the schema contains an account's parent or sponsoring account.

To change a schema's status, edit the **STATUS** entries in the **config_dist.conf** file and then use the **load_config_dist** utility to load the distribution information into the primary database schema.

> **Note:**
>
> The **load_config_dist** utility overwrites existing distributions. If you are updating distributions, you cannot load new distributions only. You must load complete sets of distributions each time you run the **load_config_dist** utility.

To change a schema's status, perform the following on the primary installation machine:

1. Go to the *BRM_home***/apps/multi_db** directory and open the **config_dist.conf** file.
2. Change the values in the **STATUS** entries:

   > **Note:**
   >
   > If your system contains multiple secondary database schemas, create a new set of entries for each secondary database schema.

   ```
   DB_NO = "0.0.0.1" ;              # 1st database config. block
   PRIORITY = 1 ;
   MAX_ACCOUNT_SIZE = 100000 ;
   STATUS = "OPEN" ;
   SCHEMA_NAME = "pin111x" ;

   DB_NO = "0.0.0.2" ;              # 2nd database config. block
   PRIORITY = 3;
   MAX_ACCOUNT_SIZE = 50000 ;
   STATUS = "OPEN" ;
   SCHEMA_NAME = "pin112x" ;
   ```

3. Save and close the file.
4. Make sure the **pin_config_distribution** utility is not running.

   See "pin_config_distribution".
5. From the *BRM_home***/apps/multi_db** directory, run the **load_config_dist** utility.

   See "load_config_dist".
6. Stop and restart all Connection Managers (CMs).

   > **Note:**
   >
   > To check how full your schemas are, see "Monitoring Database Space" in *BRM Installation Guide*.

Setting Database Schema Priorities

Database schema priority determines when customer accounts are created in a particular schema relative to other schemas. Multidatabase Manager assigns accounts to an open schema with the highest priority.

If all schemas have the same priority, Multidatabase Manager chooses an open schema at random in which to create the account. This distributes accounts evenly across all schemas. However, BRM locates accounts as follows:

- All accounts with nonpaying child units in the same schema as their paying parent bill units
- All sponsored accounts in the same schema as their sponsoring accounts

> **Note:**
>
> To limit the number of accounts in your primary database schema, set your primary database schema to a *lower* priority than the secondary database schemas. Accounts will be created in the secondary database schemas when possible.

To change schema priorities, edit the **PRIORITY** entries in the **config_dist.conf** file and then use the **load_config_dist** utility to load the distribution information into the primary database schema.

> **Note:**
>
> The **load_config_dist** utility overwrites all distributions already in the database. When adding or updating distributions, beware that you cannot load only new and changed distributions.

To change schema priorities, do the following on the primary installation machine:

1. Go to the *BRM_home***/apps/multi_db** directory, and open the **config_dist.conf** file.
2. Edit the **PRIORITY** entries. In the following example, BRM creates accounts on schema 0.0.0.2 because it has the highest priority setting of all open schemas.

> **Note:**
>
> If your system contains multiple secondary database schemas, create a new set of entries for each secondary database schema.

```
DB_NO = "0.0.0.1" ;              # 1st database config. block
PRIORITY = 1 ;
MAX_ACCOUNT_SIZE = 100000 ;
```

```
STATUS = "OPEN" ;
SCHEMA_NAME = "pin111x" ;

DB_NO = "0.0.0.2" ;             # 2nd database config. block
PRIORITY = 3;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN" ;
SCHEMA_NAME = "pin112x" ;

DB_NO = "0.0.0.3" ;             # 3rd database config. block
PRIORITY = 5;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "CLOSED" ;
SCHEMA_NAME = "pin113x" ;
```

3. Save and close the file.
4. Make sure the **pin_config_distribution** utility is not running.

   See "pin_config_distribution".
5. From a command prompt, go to the *BRM_home***/apps/multi_db** directory and run the **load_config_dist** utility.

   See "load_config_dist".
6. Stop and restart all CMs.

> **Note:**
>
> To check how full your schemas are, see "Monitoring Database Space" in *BRM Installation Guide*.

Creating Custom Tables That Are Available to All Database Schemas

You can create or modify custom tables after your multischema system is installed and configured.

To make those tables available to your secondary database schemas:

1. Create or modify your custom table in the primary database schema by connecting to the CM on the primary installation machine.
2. Name your table by using the following naming conventions:
   - **CONFIG***
   - **PLAN***
   - **PRODUCT***
   - **RATE***
   - **UNIQUEN***
   - **STRINGS***
   - **CHANNEL***

- **SEARCH***
- **ZONE*_T**
- **TOD***
- **FOLD***

3. Run the **pin_multidb.pl -i** script as follows:
   a. Go to the *BRM_home*/**setup/scripts** directory, and run **pin_multidb.pl -i**:

      ```
      cd BRM_home/setup/scripts
      perl pin_multidb.pl -i
      ```

   b. At the following prompt, enter **y** to begin configuration:

      ```
      Do you want to start the configuration now? (y/n):   y
      ```

   c. At the following prompt, enter **2** to initialize the primary database schema:

      ```
      Please enter the starting step (0-8). If you don't know, enter 0:   2
      ```

   d. Exit the **pin_multidb.pl** script.
4. Run **pin_multidb.pl -R** to re-create all refresh groups in the secondary database schemas:

   ```
   cd BRM_home/setup/scripts
   perl pin_multidb.pl -R group
   ```

   where *group* is the name of the group your table belongs to.

For more information, see "pin_multidb".

Synchronizing Database Schema Data Dictionaries

During normal multischema operations, the data dictionaries of the primary and secondary database schemas are updated automatically if you make the changes using the Storable Class Editor.

For information about adding, deleting, and modifying storable classes in the data dictionary, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

If there is a failure in making the changes to any of the secondary database schemas, the secondary database schema's data dictionary is in an inconsistent state with respect to the primary database schema. The failure is reported.

To synchronize the secondary database schema's data dictionary with the primary database schema's data dictionary, do the following on the primary installation machine:

1. Go to the *BRM_home*/**apps/multi_db** directory, and open the **pin.conf** file.
2. Verify all entries for the primary database schema.
3. Go to the *BRM_home*/**bin** directory, and run the **multi_db_synch_dd** script.

   If an error occurs, the application reports it.

Synchronizing the Database Schema /uniqueness Objects

During normal multischema operations, the **/uniqueness** objects in the primary and secondary database schemas are updated automatically. BRM uses this object in a multischema environment to locate subscribers. It contains a cache of services and must stay synchronized with the service cache in the primary database schema.

> **Note:**
>
> To determine whether the **/uniqueness** object in a secondary database schema is out of synchronization, use **sqlplus** to compare the entries in the **uniqueness_t** database table with those in the **service_t** database table. There should be a one-to-one relationship.

If the database tables are not synchronized, run the **load_pin_uniqueness** utility on the secondary database schemas to update the **/uniqueness** object with the current service data (see "load_pin_uniqueness"). This utility overwrites existing **/uniqueness** objects in the schemas.

1. Go to the *BRM_home***/apps/multi_db** directory.
2. Make sure the **pin_multidb** utility is not running. See "pin_multidb".

   The **pin_multidb** utility calls the **load_pin_uniqueness** utility when you configure a multischema environment. Therefore, you must stop the **pin_multidb** utility before you run the **load_pin_uniqueness** utility.
3. Use the following command to run the **load_pin_uniqueness** utility:

   ```
   load_pin_uniqueness
   ```
4. Stop and restart the CM.
5. Verify that the **/uniqueness** object was loaded by using one of the following to display the **/uniqueness** object:
   - Object Browser.
   - **robj** command with the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Changing the Interval for Checking New Accounts

When a multischema system is installed, a time interval is set to check for new accounts in the login table and to verify that the accounts were fully created.

To change the interval:

1. Save a copy of *BRM_home***/apps/multi_db/pin.conf**, and open the original file in a text editor.
2. Set the desired frequency value for the **pin_confirm_logins** utility in minutes:

   ```
   - pin_confirm_logins     frequency     10
   ```

   See the documentation in the **pin.conf** file for information about the frequency value.

See "pin_confirm_logins" for information about the utility.

3. Save and close the file.
4. Restart **pin_confirm_logins**.

Changing the Interval for Updating the Distribution Table

A time interval for updating the distribution table is originally set when the multischema system is installed. You can change the interval by running the **pin_config_distribution** utility on a different schedule. This utility governs the number of accounts in each schema.

To change the interval:

1. Save a copy of *BRM_home***/apps/multi_db/pin.conf**, and open the original file in a text editor.
2. Set the desired frequency value for the **pin_config_distribution** utility in hours:

   ```
   - pin_config_distribution     frequency     10
   ```

   See the documentation in the **pin.conf** file for information about the frequency value.

   See "pin_config_distribution" for information about the utility.
3. Save and close the file.
4. Restart **pin_config_distribution**.
5. Verify that the utility was successful by using one of the following to display the **/config/distribution** object:
   - Object Browser.
   - **robj** command with the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

49
Multischema Utilities

Learn how to configure and manage the schemas in the Oracle Communications Billing and Revenue Management (BRM) database by using the multischema utilities.

Topics in this document:

load_config_dist

Use this utility to load the configuration/distribution table for database schema priorities into a BRM multischema system. You define the schema priorities in the *BRM_home***/apps/multi_db/config_dist.conf** file.

> **Note:**
>
> - You must stop the "pin_config_distribution" utility before you run the **load_config_dist** utility.
> - The **load_config_dist** utility overwrites the existing schema priorities table. If you are updating schema priorities, you cannot load new priorities only. You must load the complete schema priorities table each time you run the **load_config_dist** utility.
> - To connect to the BRM database, the **load_config_dist** utility needs a configuration file in the directory from which you run the utility.

Location

*BRM_home***/bin**

Syntax

```
load_config_dist
```

Parameters

There are no input parameters for the **load_config_dist** utility.

Results

The progress of the program is displayed on the screen.

> **Note:**
>
> You must restart the Connection Manager (CM) to make new schema priorities available.

load_pin_uniqueness

Use this utility to update the **/uniqueness** object in a BRM multischema system when the object is not synchronized with the **/service** object in the schema.

> **Note:**
>
> - The "pin_multidb" utility calls this utility to create a **/uniqueness** object from a **/service** object in a multischema environment; therefore, stop the **pin_multidb** utility before you run the **load_pin_uniqueness** utility.
> - The **load_pin_uniqueness** utility overwrites existing **/uniqueness** objects. If you are updating **/uniqueness** objects, you cannot load new **/uniqueness** objects only. You must load complete sets of **/uniqueness** objects each time you run the **load_pin_uniqueness** utility.
> - To connect to the BRM database, the **load_pin_uniqueness** utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/**bin**

Syntax

```
load_pin_uniqueness
```

Parameters

There are no input parameters for the **load_pin_uniqueness** utility.

Results

The progress of the program is displayed on the screen.

> **Note:**
>
> You must restart the CM to make the **/uniqueness** updates available.

pin_config_distribution

Use this utility to update the multischema configuration object (the **/config/distribution** object). You can update the configuration at regular intervals by running the utility with the **cron** command.

To run **pin_config_distribution** directly, edit *BRM_home***/apps/multi_db/pin.conf** and run the command from that directory.

> **Note:**
>
> For multischema systems, you must run the utility separately against each schema in your system.

Location

*BRM_home***/bin**

Syntax

```
pin_config_distribution
```

Parameters

There are no input parameters for the **pin_config_distribution** utility.

Results

The progress of the program is displayed on the screen.

pin_confirm_logins

Use this utility to update the **/uniqueness** object in the primary BRM database schema with entries that have been confirmed and to delete logins that are not found.

> **Note:**
>
> If you use dblinks for managing the uniqueness table in a multischema environment, this utility is not required.

Define the frequency for running this utility in the *BRM_home***/apps/multiDB/pin_multidb.conf** file.

The **pin_multidb** script with the **-f** parameter starts **pin_confirm_logins**, but you can run the command directly if it is terminated or if you need to restart the CM after reconfiguration.

Location

*BRM_home***/apps/multiDB**

Syntax

```
pin_confirm_logins
```

Parameters

There are no input parameters for the **pin_confirm_logins** utility.

Results

The progress of the program is displayed on the screen.

pin_mta_monitor

Use this utility to monitor and regulate the thread activity of a BRM multithreaded application (MTA) without interrupting the application. You must start the MTA before you run the **pin_mta_monitor** utility.

> **Note:**
>
> To connect to the BRM database, **pin_mta_monitor** uses the configuration file of the MTA that it monitors. For example, if you use **pin_mta_monitor** to track the thread activity of **pin_inv_export**, **pin_mta_monitor** uses the **pin.conf** file of **pin_inv_export**. See "Configuring Your Multithreaded Application" in *BRM Developer's Guide*.
>
> You must run **pin_mta_monitor** from the *same* directory from which you run the MTA.

Location

BRM_home/**bin**

Syntax

```
pin_mta_monitor [mta_application]
(mon)> command
```

Parameters

This utility runs in an interactive mode. In this mode, you enter single-letter commands to perform individual actions. To run **pin_mta_monitor** for an MTA, you must be at the monitor (**(mon)>**) prompt.

Use the following command to enter into the monitor prompt:

```
pin_mta_monitor [mta_application]
```

where *mta_application* is the MTA that **pin_mta_monitor** tracks.

The **(mon)>** prompt appears.

Commands

Table 49-1 lists the commands that you can enter at the monitor prompt.

Table 49-1 pin_mta_monitor Commands

Command	Description
p	Starts monitoring *mta_application*. The utility prints the thread activity for *mta_application* to **stdout**.
t [+|-] *number*	Regulates the thread load on your MTA utility, where *number* specifies the number by which you want to increase or decrease the number of threads that *mta_application* uses. • **t [+]** *number* adds the specified number of threads to the thread pool of *mta_application*. • **t [-]** *number* removes the specified number of threads from the thread pool of *mta_application*.
?	Displays the **pin_mta_monitor** utility's commands.
q	Stops monitoring *mta_application* and exits the utility.

Results

This utility logs messages to **stdout**.

pin_multidb

Use this script to configure a BRM multischema system. You use this script for the initial configuration of a new multischema system and to configure additional secondary schemas when necessary.

> **Note:**
>
> - *Before* running **pin_multidb.pl**, edit the *BRM_home***/apps/multiDB/pin_multidb.conf** file.
> - Run **pin_multidb.pl** on the machine that hosts the CM and Data Manager (DM) of the multischema system.

For more information about installing a multischema BRM system, see "Installing a Multischema System" in *BRM Installation Guide*.

Location

*BRM_home***/setup/scripts**

Syntax

```
pin_multidb [-i] [-f] [-R all | group] [-r group] [-h]
```

Parameters

> **Note:**
>
> To fully configure a multischema system, run the utility with **-i** and then run it again with **-f**.

-i
Initializes the primary and secondary schemas.

-f
Finalizes the multischema installation.

-R all | *group*
Re-creates all refresh groups or a specified refresh group.

- **-R all**: Re-creates all refresh groups.

  > **Note:**
  >
  > After you run **pin_multidb.pl -R all**, run the **create_procedures**_*character_set*.**plb** script and the **grant_permissions_oracle.plb** script, which sets up the schema qualifications for payment processing, where *character_set* specifies the database character set of either **UTF8** or **AL32UTF8**.
  >
  > See "Setting Up Schema Qualifications" in *BRM Installation Guide*.

- **-R** *group*: Re-creates the specified refresh group, where *group* can be any of the values listed in Table 49-2.

-R Group Value	Description
CONFIG	Configuration objects
PRICE	Pricing objects
UNIQUENESS	Uniqueness objects
GENERAL	General objects

  The refresh frequencies are specified in the **pin_multidb.conf** file.

-r *group*
Forces a refresh of the specified group of objects, where *group* can be any of the values listed in Table 49-3.

-r Group Value	Description
CONFIG	Configuration objects
PRICE	Pricing objects
UNIQUENESS	Uniqueness objects

-r Group Value	Description
GENERAL	General objects

The refresh frequencies are specified in the **pin_multidb.conf** file.

-h
Displays the syntax and parameters for this utility.

Results

If the **pin_multidb** script does not notify you that it was successful, check that the data in both schemas are the same, or look in the utility log file (**pin_multidb.log**) 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.

Part VIII
Running Business Operations

This part describes how to configure Oracle Communications Billing and Revenue Management (BRM) Business Operations Center. It contains the following chapters:

- Using Business Operations Center
- Business Operations Center Utilities

50

Using Business Operations Center

Learn how to set up, manage, and use Oracle Communications Billing and Revenue Management (BRM) Business Operations Center.

Topics in this document:

Before you read this document, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

About Using Business Operations Center

You use Business Operations Center to create, schedule, and view the results of the following operations:

- **Billing:** Finds accounts that need to be billed; calculates the balance due for each bill unit in the accounts, including all usage and cycle fees; and creates a bill for the balance due.
- **Payment Collection**: Collects the balance due for accounts that use the payment card (credit or debit card) and direct debit (cash) payment methods.
- **Invoicing:** Generates invoices that list the events that were charged for, and the customer's total account balance.
- **Generating General Ledger Reports:** Generates general ledger (G/L) reports to collect G/L data for G/L accounts.
- **Refunding payments:** Finds accounts that have refund items, and makes online refund transactions.
- **Synchronizing product catalogs:** Synchronizes the catalogs stored in the BRM server with updated values from customers.
- **Workflows:** Runs a set of jobs in order, enabling an automated end-to-end billing process. You can choose to run two or more of these operations in order: billing, invoicing, payments, and refunds.
- **Your Custom Applications:** Runs your custom applications and displays the results in the Business Operations screen. You must configure Business Operations Center to run

your custom application. See "Defining a Custom Category" in *Business Operations Center Online Help* for more information.

- **Viewing Business Trends:** Shows business trends based on data generated by the preceding operations.

If you have a multischema BRM system, you can specify whether to run a business operations job against the primary schema, a secondary schema, or all schemas. By default, jobs are run against all schemas.

For more information about running a job, see "Running Business Operations Jobs" in *Business Operations Center Online Help*.

> **Note:**
>
> - Starting with BRM 12.0.0.3.0 with Interim Patch 31426340, Business Operations Center calls the PCM_OP_JOB_EXECUTE opcode to run the BRM utilities for these operations. For more information, see "How Business Operations Center Runs Jobs".
> - In BRM 12.0 Patch Set 3 and earlier releases, Business Operations Center calls the **pin_job_executor** utility to run the BRM utilities for these operations. For more information, see "pin_job_executor".

About Running Jobs in Business Operations Center

When running jobs in Business Operations Center, Oracle recommends the following:

- Because the jobs that Business Operations Center can run often depend on other processes that Business Operations Center cannot run, determine whether you must supplement some of the jobs run in Business Operations Center by running additional processes elsewhere in BRM.

  For example, to process off-cycle (daily, weekly, quarterly, and so on) charges, run the **pin_cycle_fees** utility as a custom job in Business Operations Center, from the BRM command line, as part of a daily cron job. Otherwise, a situation such as the following may occur:

  Account A has a quarterly (three-month) billing cycle and receives 100 free minutes each month. The free minutes are granted by a monthly cycle forward fee, and they do not carry over to the following month. When Business Operations Center runs billing (**pin_bill_accts**), all pending cycle forward fees are applied. But for Account A, Business Operations Center will run billing only once every three months. At that time, three monthly cycle forward fees will be pending for Account A. Billing will apply them as follows:

  – Two past monthly cycle forward fees will grant Account A a total of 200 expired free minutes.

  – One current monthly cycle forward fee will grant Account A 100 free minutes that will be valid for one month.

  Hence, if you do not process the monthly cycle forward fees that do not align with Account A's quarterly billing cycle, when Business Operations Center runs billing for the account once every three months and processes the two past-due monthly

grants of free minutes at that time, those minutes will be unusable because they expire one month after coming due.

- Do not run the BRM billing scripts (**pin_bill_day**, **pin_bill_week**, **pin_bill_month**) because they automatically run the same utilities (**pin_bill_accts**, **pin_collect**, **pin_inv_accts**, and so on) that Business Operations Center runs.

For more information about running jobs, see "Running Business Operations Jobs" in *Business Operations Center Help*.

How Business Operations Center Runs Jobs

> **Note:**
>
> Business Operations Center calls PCM_OP_JOB_EXECUTE for only BRM 12.0.0.3.0 Patch 31426340 or later releases. If you are using BRM 12.0 Patch Set 3 or an earlier release, Business Operations Center calls the **pin_job_executor** utility directly. See "pin_job_executor".

Business Operations Center runs jobs by doing the following:

1. Calling the PCM_OP_JOB_PROCESS_TEMPLATE opcode to create a **/job_template** object, which specifies the job details such as the type of business operations job to run and the database schema on which to run it. See "Managing Business Operations Job Templates" in *BRM Opcode Guide*.
2. Calling the PCM_OP_JOB_EXECUTE opcode to run applications in either a job context or a non-job context, depending on the PIN_FLD_FLAGS input flist setting:
   - **1**: Opens a job context and then internally runs a business operations utility against the target schema. This setting is used when running any business operations job. See "Running Business Operations Jobs" in *BRM Opcode Guide* for more information.
   - **0**: Runs applications in a non-job context. This setting is used when running **pin_virtual_time** or a custom application. See "Running BRM Applications" in *BRM Opcode Guide* for more information.

Setting Up Custom Clients to Run Business Operations and BRM Applications

> **Note:**
>
> Perform this step only if you are using BRM 12.0.0.3.0 with Interim Patch 31426340 or later releases.

You can set up a custom client application to run business operation jobs or other BRM applications. To do so, you must:

1. Configure PCM_OP_JOB_EXECUTE to run BRM applications. See "Configuring PCM_OP_JOB_EXECUTE to Run BRM Applications".
2. Configure the **pin_job_executor** process to listen for calls from PCM_OP_JOB_EXECUTE. See "Configuring pin_job_executor to Listen for Opcode Calls".
3. If you upgraded from a previous release, do this:
   - Remove the **PASSWORDLESS_SSH_USER** property from the *Domain_home***/lib/bocws-config.properties** file, where *Domain_home* is the WebLogic Server domain home directory in which Business Operations Center is deployed.
   - Remove the passwordless SSH set up between BRM Server and Business Operations Center.
4. Customize your client application to call the PCM_OP_JOB_PROCESS_TEMPLATE opcode to create or modify a **/job_template** object, which specifies job details such as the type of business operations job to run and the schema on which to run it. See "Managing Business Operations Job Templates" in *BRM Opcode Guide*.
5. Customize your client application to call the PCM_OP_JOB_EXECUTE opcode to do the following:
   - Run business operations jobs. See "Running Business Operations Jobs" in *BRM Opcode Guide*.
   - Run any other BRM applications. See "Running BRM Applications" in *BRM Opcode Guide*.
6. Configure Business Operations Center to not render invoices in BI Publisher. See "Rendering Invoices in a Third-Party Invoice Application".

Configuring PCM_OP_JOB_EXECUTE to Run BRM Applications

The PCM_OP_JOB_EXECUTE opcode internally calls the **pin_job_executor** process, which in turn calls the BRM application that you want to run.

To configure the PCM_OP_JOB_EXECUTE opcode to run BRM applications:

1. Open the *BRM_home***/apps/pin_job_executor/Infranet.properties** file in a text editor.
2. Set the following entry to match your environment:

   ```
   infranet.server.portNr=Port
   ```

   where *Port* is the port number on which **pin_job_executor** is listening for calls from the CM.
3. Change the value of the **infranet.pcp.ssl.wallet.location** entry from **$PIN_HOME/wallet/client** to **$PIN_HOME/wallet/server**:

   ```
   infranet.pcp.ssl.wallet.location=$PIN_HOME/wallet/server
   ```
4. Add the following entry:

   ```
   infranet.opcode.handler.JOB_EXECUTE=com.oracle.communications.brm.pje.JobOpcodeHandler
   ```
5. Add the following entry:

```
infranet.pje.executable.locations=bin,apps,subDirectory
```

where *subDirectory* is a directory under **$PIN_HOME**. The PCM_OP_JOB_EXECUTE opcode runs applications only if they reside in **$PIN_HOME/bin**, **$PIN_HOME/apps**, or a **$PIN_HOME** subdirectory that you specify.

6. Specify the number of threads that the opcode can spawn to run applications on multischema systems:

   ```
   infranet.threadpool.size=ThreadNumber
   ```

   where *ThreadNumber* is the number of threads that can be run. The default is 5, but you can increase it to improve performance.

7. Ensure that all applications that PCM_OP_JOB_EXECUTE will run, including any custom applications, are located in **$PIN_HOME/bin**, **$PIN_HOME/apps**, or **$PIN_HOME/***subDirectory*.

8. Save and close the file.

9. Stop and restart the **pin_job_executor** utility by running the following commands from the *BRM_home***/bin** directory:

   ```
   .\stop_pje
   .\start_pje
   ```

Configuring pin_job_executor to Listen for Opcode Calls

You must configure **pin_job_executor** to run as a background process, listening at a configured port for an opcode call from PCM_OP_JOB_EXECUTE. To do so:

1. Open the *BRM_home***/sys/cm/pin.conf** file in a text editor.

2. Add the following entry:

   ```
   - cm em_group pje PCM_OP_JOB_EXECUTE
   ```

3. Set the following entry to match your environment:

   ```
   - cm em_pointer pje hostname port
   ```

   where:

   - *hostname* is the name or IP address of the server on which **pin_job_executor** is running.
   - *port* is the port on which **pin_job_executor** is listening.

4. Save and close the file.

5. To log audit information about commands that are run in a non-job context, do this:

   a. Open the *BRM_home***/sys/data/config/pin_notify** file in a text editor.

   b. Add the following entry:

      ```
      179 0 /event/activity/job_request
      ```

   c. Save and close the file.

   d. Load the **pin_notify** file into the BRM database:

      ```
      load_pin_notify -v pin_notify
      ```

See "load_pin_notify" in *BRM Managing Customers* for more information.

e. Stop and restart the CM.

6. Stop and restart the **pin_job_executor** utility by running the following command from the *BRM_home***/bin** directory:

```
.\start_pje
```

Rendering Invoices in a Third-Party Invoice Application

By default, when Business Operations Center generates invoices, it renders the invoices using Oracle Business Intelligence (BI) Publisher. If you use a third-party invoice application, you must configure Business Operations Center to not render invoices in BI Publisher.

To prevent Business Operations Center from rendering invoices in BI Publisher:

1. Open the *BRM_home***/apps/pin_job_executor/Infranet.properties** file in a text editor.

> **Note:**
>
> Only administrators with write permissions can make changes to the **Infranet.properties** file.

2. Search for the following line:

```
- infranet.job.rendering=true
```

3. Change **true** to **false**.

```
infranet.job.rendering=false
```

Invoices will not be rendered in BI Publisher.

4. Save and close the file.

5. Stop and restart the **pin_job_executor** utility by running the following commands from the *BRM_home***/bin** directory:

```
.\stop_pje
.\start_pje
```

Setting Up Business Operations Center to Run Custom Applications on Multischema Systems

You can set up Business Operations Center to run your custom application. To do so, you use the Business Operations Center GUI to create a custom category that defines the name and location of your script as well as the parameters associated with the script. For more information, see "Setting Up Custom Categories" in *Business Operations Center Online Help*.

If you want Business Operations Center to run your custom application on a BRM multischema system, you must also customize the application to do the following:

- Accept the **-job_id** *idNumber* and **-schema** *schemaNumber* command-line arguments, where *idNumber* is the job ID and *schemaNumber* is the database schema on which to run the application.

  Business Operations Center passes these two arguments with the command to run your application. For example, if the command-line syntax for running your application is **my_custom_app -f file**, Business Operations Center would run the following command:

  my_custom_app -f file -job_id *idNumber* **-schema** *schemaNumber*

- Build the POID of the **/job/boc** database object and then return it in its output to Business Operations Center. The format for building the POID is:

  0.0.0.*schemaNumber* **/job/boc** *idNumber*

  For example, the **/job/boc** POID for the following command would be **0.0.0.2 /job/boc 1234567**.

  ```
  my_custom_app -f file -job_id 1234567 schema 2
  ```

- Update the **/job/boc** database object with the application's status as it runs. Set the object's PIN_FLD_STATUS field to one of the following:
  - **#define PIN_BOC_STATUS_APP_START 2**: Indicates that the application has started.
  - **#define PIN_BOC_STATUS_APP_SUCCESS 3**: Indicates that the application completed successfully.
  - **#define PIN_BOC_STATUS_APP_FAIL 4**: Indicates that the application failed.
- For applications that use the multithreaded application (MTA) framework, read the schema number, set it in the database number of PIN_FLD_POID_VAL, and then run the job against the specified schema,

  Your application can fetch this field from **app_flistp** and then use it to extract the database number. For example:

  ```
  vp = PIN_FLIST_FLD_GET (app_flistp, PIN_FLD_POID_VAL, MTA_MANDATORY,
  ebufp);
  if ( vp ) {
     database = PIN_POID_GET_DB ((poid_t*)vp);
  }
  ```

  Also customize your application to do the following:
  - Perform searches in a BRM database schema by including the database number of PIN_FLD_POID_VAL in the **/job/boc** POID. The POID for the search flist can be created as follows:

    ```
    s_pdp = PIN_POID_CREATE(database, "/search", -1, ebufp);
    ```

  - Run an opcode against a BRM database schema by setting the database portion of the opcode's PIN_FLD_POID input flist field.

  - Create a **/process_audit/billing** object on the specified schema and then set the database portion of the object's PIN_FLD_POID field to the schema number. For example, in the following POID, the application would set only *schemaNumber*:

    ```
    0.0.0.schemaNumber /process_audit/billing 123456
    ```

- For non-MTA applications, read the **-job_id** *idNumber* and **-schema** *schemaNumer* arguments and then:
  - Run against the specified database schema
  - Create a **/process_audit/billing** object on the specified database schema

Improving Business Operations Center Performance

The PCM_OP_JOB_EXECUTE opcode internally calls the **pin_job_executor** process, which in turn calls the BRM application that you want to run. In a multischema environment, the **pin_job_executor** process runs applications on all schemas simultaneously using multiple threads. The process uses five threads by default, but you can tune the number of threads to improve performance.

To improve Business Operations Center performance:

1. Open the *BRM_home***/apps/pin_job_executor/Infranet.properties** file in a text editor.

   > **Note:**
   >
   > Only administrators with write permissions can make changes to the **Infranet.properties** file.

2. Set the following entry to the number of threads that can be spawned by the **pin_job_executor** process:

   ```
   infranet.threadpool.size=ThreadNumber
   ```

3. Save and close the file.

Enabling Secure Communication for the pin_job_executor Utility

> **Note:**
>
> Perform this step only if you are using BRM 12.0.0.3.0 or earlier releases. Starting with BRM 12.0.0.3.0 Patch 31426340, Business Operations Center calls the PCM_OP_JOB_EXECUTE opcode rather than the **pin_job_executor** utility.

By default, the ability to use Secure Sockets Layer (SSL) or Transport Layer Security (TLS) with the **pin_job_executor** utility is disabled. To enable secure communication between Business Operations Center and the Connection Manager (CM), you must configure SSL/TLS for the internal **pin_job_executor** utility.

To enable SSL for Business Operations Center:

1. Open the *BRM_home*/**apps/pin_job_executor/Infranet.properties** file in a text editor, where *BRM_home* is the directory in which BRM is installed.

   > **Note:**
   >
   > Only administrators with write permissions can make changes to the **Infranet.properties** file.

2. Search for the following line:

   ```
   - infranet.pcp.ssl.enabled=false
   ```

3. Change **false** to **true**.

   ```
   - infranet.pcp.ssl.enabled=true
   ```

4. Add the following entry:

   ```
   - infranet.pcp.ssl.wallet.location=$PIN_HOME/wallet/server
   ```

5. Add the following entry:

   ```
   - infranet.pcp.ssl.wallet.filename=wallet_name.sso
   ```

   where *wallet_name* is the name of your Oracle wallet.

6. Save and close the file.

About Generating Metrics to Display in Business Operations Center

You use the **pin_generate_analytics** utility to generate business metrics data for accounts by status, accounts by subscription, payments, billed revenue, and accounts receivable (A/R) and to load the metrics data into the BRM database. Business Operations Center displays graphs based on the business metrics data generated by the **pin_generate_analytics** utility.

You can run **pin_generate_analytics** manually or configure a scheduler, such as cron, to run it at predefined intervals. For more information, see "Generating Business Metrics Data".

The **pin_generate_analytics** utility generates and loads the following metrics into the BRM database:

- Accounts by status: Generates metrics about the number of subscribers for each status at the time the utility is run: Active (10100), Inactive (10102), Closed (10103).
- Accounts by subscription: Generates metrics about the number of subscriptions for each product at the time the utility is run based on the status of subscription: Active (1), Inactive (2), Closed (3).

- Payment: Generates metrics about payments collected based on the payment type: prepaid (10000), invoice (10001), direct debit (10002), credit card (10003), cash (10011), and check (10012).

  When you run the utility for the first time, data will be generated for the previous day. For subsequent runs, the start date is set to the last run date and the end date is set to the current date.
- Billed revenue: Generates metrics about the amount billed and the number of bills generated.

  When you run the utility for the first time, data will be generated for the previous day. For subsequent runs, the start date is set to the last run date and the end date is set to the current date.
- Accounts receivable: Generates metrics about the A/R for the last one year with respect to month and the year.

  When you run the utility, data will be generated for the previous 12 months from the date the utility is run.

Generating Business Metrics Data

You can generate metrics data for all business operations or just for specific business operations, such as subscriptions or payments collected, by running the **pin_generate_analytics** utility. For more information, see "pin_generate_analytics".

To generate metrics data for all business operations:

1. Go to the *BRM_home***/apps/pin_generate_analytics** directory.
2. Run the following command:

   ```
   pin_generate_analytics
   ```

The following sections show the commands to run for generating metrics data for specific business operations.

Generating Metrics for the Number of Subscribers

To generate business metrics data about the number of subscribers for each status at the time the utility is run, run the following command:

```
pin_generate_analytics -acc
```

Generating Metrics for the Number of Subscriptions Per Product

To generate business metrics data business metrics data about the number of subscriptions for each product based on the status of subscription at the time the utility is run, run the following command:

```
pin_generate_analytics -subs
```

Generating Metrics for Payments Collected

To generate business metrics data about the number and amount of payments collected, grouped by payment type, run the following command:

```
pin_generate_analytics -pymt
```

The first time you run this command, it generates business metrics data for the previous day. For example, if you run the command on January 2, it generates

business metrics data for January 1. For subsequent runs, the start date is set to the last run date and the end date is set to the current date.

Generating Metrics for Amount Billed

To generate business metrics data about the amount billed and the number of bills generated, run the following command:

```
pin_generate_analytics -billing
```

The first time you run this command, it generates business metrics data for the previous day. For example, if you run the command on January 2, it generates business metrics data for January 1. For subsequent runs, the start date is set to the last run date and the end date is set to the current date.

Generating Metrics for Accounts Receivable

To generate business metrics data about the account receivable for the last one year with respect to month and the year, run the following command:

```
pin_generate_analytics -acc_recv
```

Creating Indexes to Improve Performance While Generating Metrics Data

When running the **pin_generate_analytics** utility to generate metrics data, you can improve performance by creating indexes.

To create indexes to improve performance while generating metrics data:

1. Go to the *BRM_home*/**sys/dd/data** directory.
2. Run the following command, which opens SQL*Plus:

   ```
   % sqlplus login@database_alias
   Enter password: password
   ```

   where:

   - *login* is the login name to use for connecting to the BRM database.
   - *password* is the password for *login*.
   - *database_alias* is the BRM database alias.
3. Run the following command:

   ```
   SQL> @create_boc_pga_indexes.source
   ```

   The indexes are created.
4. Run the following command, which exits SQL*Plus:

   ```
   SQL> exit
   ```

51

Business Operations Center Utilities

Learn how to generate business metrics and run Oracle Communications Billing and Revenue Management (BRM) applications by using the Business Operations Center utilities.

Topics in this document:

pin_generate_analytics

Use this utility to generate business metrics data for accounts by status, accounts by subscription, payments, billed revenue, and accounts receivable (A/R) and to load the metrics data into the BRM database. Business Operations Center displays graphs based on the business metrics data generated by this utility.

You can run **pin_generate_analytics** manually or configure a scheduler, such as cron, to run it at predefined intervals. For more information, see "Generating Business Metrics Data".

> **Note:**
>
> - If you run this utility without parameters, data will be generated for all operations: accounts, subscription, payment, billed revenue, and A/R.
> - For each operation, run this utility only once per day. If you rerun the utility for an operation on the same day, the parameter is ignored and data will not be generated.
> - You can improve the utility's performance by creating indexes in the BRM database. See "Creating Indexes to Improve Performance While Generating Metrics Data".

Location

BRM_home/**apps/pin_generate_analytics**

Syntax

```
pin_generate_analytics [-help] [-verbose] [-acc | -subs | -pymt | -billing | -acc_recv]
```

Parameters

-help
Displays the syntax and parameters for this utility.

-verbose
Displays information about successful or failed processing as the utility runs.

-acc
Generates business metrics data about the number of subscribers for each status at the time the utility is run.

-subs
Generates business metrics data about the number of subscriptions for each product based on the status of subscription at the time the utility is run.

-pymt
Generates business metrics data about payments collected based on the payment type.
When you run the utility for the first time with this parameter, business metrics data is generated for the previous day. For example, if you run the utility on January 2, business metrics data are generated for January 1.
For subsequent runs, the start date is set to the last run date and the end date is set to the current date.

-billing
Generates business metrics data about the amount billed and the number of bills generated.
When you run the utility for the first time with this parameter, business metrics data is generated for the previous day. For example, if you run the utility on January 2, business metrics data are generated for January 1.
For subsequent runs, the start date is set to the last run date and the end date is set to the current date.

-acc_recv
Generates business metrics data about A/R for the previous 12 months from the date the utility is run.
For example, if you run the utility on March 17, 2030, A/R business metrics data are generated for a period starting from March 17, 2029 until March 17, 2030.

Results

The progress of the utility is displayed on the screen.

pin_job_executor

Use this utility to run BRM applications, such as **pin_bill_accts**, **pin_collect**, or **pin_inv_accts**, and your custom MTA applications.

> **Note:**
>
> - Do not run this utility from the command line.
> - In BRM 12.0.0.3.0 and earlier, the **pin_job_executor** utility is run internally by Business Operations Center.
> - In BRM 12.0.0.3.0 Patch 31426340 and later, use the PCM_OP_JOB_EXECUTE opcode instead of this utility.

For more information, see "Using Business Operations Center".

Location

BRM_home/apps/pin_job_executor

Syntax

```
pin_job_executor  -job_template job_template_ID -request_id request_ID
                  [-start time] -end time [-job_id job_id] [-help]
```

Parameters

-job_template *job_template_ID*
Specifies the ID of the job template to run.

-request_id *request_ID*
Specifies the job request ID to use.

-start *time*
Specifies the start time of the job.

-end *time*
Specifies the end time of the job.

-job_id *job_id*
Re-runs the jobs based on the specified job ID.

-help
Displays the syntax and parameters for this utility.

Results

The progress of the program is displayed in Business Operations Center.

Part IX
Configuration File Reference

This part provides reference information about Oracle Communications Billing and Revenue Management (BRM) configuration files. It contains the following chapters:

- Business Logic pin.conf Reference
- System Administration pin.conf Reference
- business_params Reference

52

Business Logic pin.conf Reference

Learn how to use the business logic **pin.conf** settings to configure Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

See also "business_params Reference" and "System Administration pin.conf Reference".

Account Creation pin.conf Entries

Table 52-1 lists the account creation **pin.conf** entries.

Table 52-1 Account Creation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
allow_active_service_with_inactive_account	**fm_cust**	Specifies whether services can be activated (during a **SET_STATUS** operation) if the account or bill unit is inactive.	CM	Cached by the CM. Restart the CM after changing this entry.
cc_checksum	**fm_cust_pol**	Specifies whether to run checksum validation on the customer's credit card during account creation.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

Table 52-1 (Cont.) Account Creation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
config_dir	**fm_cust_pol**	Specifies the location of the ISP configuration data, which is stored in the **default.config** file. Used by the PCM_OP_CUST_POL_GET_CONFIG policy opcode.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
country	**fm_cust_pol**	Specifies the default country for new accounts (default is **USA**).	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
currency	**fm_cust_pol**	Specifies the default currency for new accounts.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
domain	**fm_cust_pol**	Specifies the email domain assigned to customers during account creation.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
new_account_welcome_msg	**fm_cust_pol**	Specifies whether the system should send the default welcome message on account creation.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
welcome_dir	**fm_cust_pol**	Specifies the location of the welcome message sent to customers after account creation.	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

Accounts Receivable pin.conf Entries

Table 52-2 lists the Accounts Receivable **pin.conf** entries.

Table 52-2 Accounts Receivable pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
calc_cycle_from_cycle_start_t	**fm_bill**	Specifies whether to calculate charge offer fees based on the charge offer's purchase date (PIN_FLD_CYCLE_START_T). See "Calculating Cycle Fees for Backdating" in *BRM Configuring and Running Billing*.	CM	This value is read by the utility when it runs. You do not need to restart the CM.
cycle_tax_interval	**fm_bill**	Determines whether deferred taxes are calculated separately for a paying parent bill unit and its nonpaying child bill units or are consolidated into a single tax item for both the parent and child bill units.	CM	Cached by the CM. Restart the CM after changing this entry.
item_search_batch	**fm_bill**	Specifies the number of items returned by a step search.	CM	This value is read by the utility when it runs. You do not need to restart the CM.
overdue_tolerance	**fm_bill**	Specifies how BRM treats amounts applied to the item when they are less than the amount due as a result of euro and Economic and Monetary Union (EMU) conversions. See "Rounding Errors for Overpayments and Underpayments" in *BRM Managing Customers*.	CM	This value is read by the utility when it runs. You do not need to restart the CM.
underdue_tolerance	**fm_bill**	Specifies how BRM treats amounts applied to the item when they are more than the amount due as a result of euro and EMU conversions. See "Rounding Errors for Overpayments and Underpayments" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.

Billing pin.conf Entries

Table 52-3 lists the Billing **pin.conf** entries.

Table 52-3 Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
actg_dom	**fm_cust_pol**	Used during account creation to determine the day of the month to run billing. See "Setting the Default Accounting Day of Month (DOM)" in *BRM Configuring and Running Billing*.	CM Billing utilities	This value is read by the utility when it runs. You do not need to restart the CM.
actg_type	**fm_cust_pol**	Specifies the default accounting type. See "Setting the Default Accounting Type" in *BRM Configuring and Running Billing*.	CM Billing utilities	This value is read by the utility when it runs. You do not need to restart the CM.
advance_bill_cycle	**fm_bill**	Sets the first billing date to be the day after account creation. See "Setting the First Billing Cycle to the Day after Account Creation" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
attach_item_to_event	**fm_act**	Specifies how BRM assigns event and service combinations to bill items. See "About Using Event and Service Combinations to Assign Bill Items" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
bill_when	**fm_cust_pol**	Specifies the default billing-cycle length. See "Setting the Default Billing-Cycle Length" in *BRM Configuring and Running Billing*.	CM	The new value becomes effective immediately. You do not have to restart the CM.
billing_segment_config_refresh_delay	**fm_cust**	Specifies how often data in the cached **/config/billing_segment** object is automatically refreshed from the database. See "Updating Billing Segments" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
config_billing_cycle	**fm_bill**	Specifies how long after the billing cycle ends that new events are considered for the previous month's bill.	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.

Table 52-3 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
config_billing_delay	**fm_bill**	Specifies the billing delay interval during which both old events (for the previous cycle) and new events (for the current cycle) can be processed. When specified, the system creates the next bill object (for the next billing cycle). Bill total calculation occurs only outside of this delay interval. See "Setting Up Delayed Billing" in *BRM Configuring and Running Billing*.	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
custom_bill_no	**fm_bill**	Specifies the accounting cycle (first or last) in which to assign a bill number to a bill in a multi-month billing cycle. See "Specifying When to Apply Custom Bill Numbers" in *BRM Configuring and Running Billing*.	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
cycle_delay_align	**fm_bill**	Specifies whether to align the charge offer purchase, cycle, and usage start and end times with the accounting cycle. See "Aligning Account and Cycle Start and End Times" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
cycle_delay_use_special_days	**fm_bill**	Sets the delayed cycle start date to the 1st of the following month for all bundles purchased on the 29th, 30th, or 31st. See "Setting Delayed Cycle Start Dates to the 29th, 30th, or 31st" in *BRM Configuring and Running Billing*.	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
deadlock_retry_count	**fm_bill_accts**	Specifies the number of retries to attempt when a deadlock occurs during a billing run. See "Specifying the Number of Retries in Case of a Deadlock".	**pin_billd**	This entry is read by the utility when it runs. You do not need to restart the CM.
delay_cycle_fees	**fm_bill**	In systems set up for delayed billing, specifies when to apply cycle forward fees and cycle rollovers (during partial billing or final billing). See "Specifying When to Apply Cycle Forward Fees and Cycle Rollovers" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-3 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
delta_step	**pin_billd**	Reduces contention at the database level during billing. See "Rearranging Accounts to Improve Billing Performance".	**pin_billd**	This entry is read by the utility when it runs. You do not need to restart the CM.
enable_30_day_proration	**fm_bill**	Allows you to base monthly proration on 30-day months rather than the actual number of days in a billing cycle. This is often needed when working in parallel with older legacy billing systems. See "About 30-Day-Based Proration" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
enforce_billing	**pin_bill_accts**	Enforces partial billing from the billing application inside the billing delay interval. See "Enforcing Partial Billing in the Billing Delay Interval" in *BRM Configuring and Running Billing*.	**pin_billd**	This entry is read by the utility when it runs. You do not need to restart the CM.
group_by_account	**pin_deferred_act**	Specifies to use one thread to process scheduled actions for one account. See "Ensuring the Sequence of Scheduled Actions".	**pin_billd**	This entry is read by the utility when it runs. You do not need to restart the CM.
keep_cancelled_products_or__discounts	**fm_subscription_pol**	Specifies whether to keep canceled charge offers and discount offers. See "Providing Discounts to Closed Accounts" in *BRM Configuring and Running Billing*.	CM	The new value becomes effective immediately. You do not need to restart the CM.
num_billing_cycles	**fm_subs**	Specifies the maximum number of billing cycles allowed between the current time and the backdated event time of a backdated operation.	CM	The new value becomes effective immediately. You do not need to restart the CM.
open_item_actg_include_prev_total	**fm_bill**	Includes previous bill totals in the pending receivable value calculated during billing for accounts that use open item accounting. See "Including Previous Balances in the Current Amount Due in Open Item Accounting" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry

Table 52-3 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
purchase_fees_backcharge	**fm_bill**	Specifies which cycle (current or next) to apply the deferred purchase fees to during billing when the deferred time coincides with the billing time. See "Specifying Which Billing Cycle to Assign to Deferred Purchase Fees" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry
rating_longcycle_roundup_flag	**fm_rate**	Enables rounding up for the long cycle.	CM	Cached by the CM. Restart the CM after changing this entry.
stop_bill_closed_accounts	**fm_bill**	Stops billing of closed bill units (**/billinfo** objects) when all items have zero due. See "Suspending Billing of Closed Accounts" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry
timestamp_rounding	**fm_bill**	Rounds timestamp to midnight. See "About Time-Stamp Rounding" in *BRM Configuring and Running Billing*. See also "About Sub-Balances" in *BRM Creating Product Offerings*.	CM	Cached by the CM. Restart the CM after changing this entry.
unset_error_status	**pin_bill_accts**	Sets billing status in a bill unit when billing errors occur. See "Setting the Bill Unit Status When Billing Errors Occur" in *BRM Configuring and Running Billing*.	**pin_billd**	This entry is read by the utility when it runs. You do not need to restart the CM.
use_number_of_days_in_month	**fm_bill**	Specifies how to calculate proration when a cycle charge offer is purchased or canceled. See "Calculating Prorated Cycle Fees" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.

Collections pin.conf Entries

Table 52-4 lists the Collections **pin.conf** entries.

Table 52-4 Collections pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
account_search_batch	**fm_collections**	Specifies the number of bill units to retrieve in collections step searches. See "Setting the Number of Bill Units Retrieved during Step Searches" in *BRM Collections Manager*.	CM	Cached by the CM. Restart the CM after changing this entry.
collections_action_dependency	**fm_collect**	Creates dependencies between collections actions in a collections scenario. See "Creating Dependencies between Collections Actions in a Scenario" in *BRM Collections Manager*.	CM	Cached by the CM. Restart the CM after changing this entry.
delivery_preference	**pin_collections_send_dunning** **pin_inv_send**	Specifies the default delivery method, email or print, for noninvoice accounts. See "Setting the Delivery Option" in *BRM Collections Manager*.	**pin_collections** **pin_inv**	The new value is effective immediately. You do not need to restart the CM.
email_body	**pin_collections_send_dunning** **pin_inv_send**	Specifies the path to a text file containing a customized message. See "Specifying a File for the Email Body" in *BRM Collections Manager*.	**pin_collections** **pin_inv**	The new value is effective immediately. You do not need to restart the CM.
execute_all_actions	**fm_collections**	Specifies whether to run all actions, including actions due before the date on which **pin_collections_process** is run for the first time, or only to run actions due after the date on which **pin_collections_process** is run for the first time. See "Performing Scheduled Collections Actions after Reinstalling Collections Manager" in *BRM Collections Manager*.	CM	The new value is effective immediately. You do not need to restart the CM.
minimum_due	**pin_collections_process**	Specifies the minimum overdue balance that a bill unit must have to qualify for collections. See "Setting the Minimum Overdue Balance to Process" in *BRM Collections Manager*.	**pin_collections**	The new value is effective immediately. You do not need to restart the CM.

Customer Management pin.conf Entries

Table 52-5 lists the Customer Management **pin.conf** entries.

Table 52-5 Customer Management pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
apply_folds	**fm_bill**	If you do not use folds, you can disable fold calculation by using this entry. This entry is on by default, that is, folds are applied, but you can turn it off to increase performance.	CM	Cached by the CM. Restart the CM after changing this entry.
apply_rollover	**fm_bill**	If you do not use rollovers, you can disable rollover calculation by using this entry. This entry is on by default, that is, rollovers are applied, but you can turn it off to increase performance.	CM	Cached by the CM. Restart the CM after changing this entry.
cycle_arrear_proration	**fm_rate**	Specifies when to prorate cycle arrears fees, at purchase or at cancellation. See "Proration for Special Cases" in *BRM Configuring and Running Billing*.	CM	Cached by the CM. Restart the CM after changing this entry.
em_db	**fm_delivery**	Specifies the Email Data Manager database number. See "Configuring the Email Data Manager" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.
intro_dir	**fm_cust_pol**	Specifies the location of the introductory message that allows a customer to confirm account creation.	CM	The new value is effective immediately. You do not need to restart the CM.

Discounting pin.conf Entries

Table 52-6 lists the Discounting **pin.conf** entries.

Table 52-6 Discounting pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
propagate_discount	**fm_subscription**	Enables immediate propagation of shared discount when a new discount is added to/deleted from the group or a member subscribes to/unsubscribes from the group. See "Configuring the Start and End Times for Discount Sharing" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.

Table 52-6 (Cont.) Discounting pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
rollover_zeroout_discounts	**fm_rate**	Zeroes-out the positive bucket. See "Configuring the Start and End Times for Discount Sharing" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.

General Ledger pin.conf Entries

Table 52-7 lists the General Ledger **pin.conf** entries.

Table 52-7 General Ledger pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
gl_segment	**fm_cust_pol**	Specifies the default G/L segment for an account during account creation. See "Changing the Default G/L Segment" in *BRM Collecting General Ledger Data*.	CM	The new value is effective immediately. You do not need to restart the CM.
transaction_grouping	**pin_ledger_report**	Specifies how many A/R accounts to group in a single transaction for **ledger_report** to run. See "Setting the Number of A/R Accounts per G/L Report" in *BRM Collecting General Ledger Data*.	**pin_billd**	The new value is effective immediately. You do not need to restart the CM.

Invoicing pin.conf Entries

Table 52-8 lists the Invoicing **pin.conf** entries.

Table 52-8 Invoicing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
database	**pin_inv_accts** **pin_inv_export** **pin_inv_send** **pin_inv_upgrade**	Specifies the database schema to which the following utilities should connect: • **pin_inv_accts** • **pin_inv_export** • **pin_inv_send** • **pin_inv_upgrade** See "Configuring the Invoice pin.conf for Multiple Database Schemas" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.

Table 52-8 (Cont.) Invoicing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
event_cache	**fm_inv**	Enables the event cache for the **PIN_FLD_BAL_IMPACTS** array.	CM	Cached by the CM. Restart the CM after changing this entry.
export_dir	**pin_inv_export**	Specifies the path to the invoice directory. See "Exporting Invoices" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.
from	**pin_inv_send**	Specifies the e-mail address of the sender. See "About Invoices" and "pin_inv_send" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.
html_template	**pin_inv_pol**	Specifies the HTML template file to use to generate invoices.	CM	The new value is effective immediately. You do not need to restart the CM.
inv_item_fetch_size	**fm_inv**	Sets the number of items to fetch when a step search is used instead of a regular search.	CM	Cached by the CM. Restart the CM after changing this entry.
invoice_db	**pin_inv_export** **pin_inv_send**	Specifies the database number for the schema. See "Configuring BRM to Use a Separate Invoice Database Schema" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.
invoice_dir	**fm_bill**	Specifies the directory where invoices are stored. See "Exporting Invoices" in *BRM Designing and Generating Invoices*.	CM	The new value is effective immediately. You do not need to restart the CM.
invoice_fmt	**pin_inv_export** **pin_inv_send**	Specifies the invoice format. See "Exporting Invoices" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.
inv_perf_features	**fm_inv**	Improves performance by removing unnecessary details from your invoices.	CM	Cached by the CM. Restart the CM after changing this entry.
sender	**fm_cust_pol** **pin_inv_send**	Specifies the name of the e-mail sender.	CM **pin_inv**	The new value is effective immediately. You do not need to restart the CM.
service_centric_invoice	**fm_inv_pol**	Enables service-centric invoicing.	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-8 (Cont.) Invoicing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
show_rerate_details	**fm_inv_pol**	Specifies whether to display shadow adjustment details on invoices. See "Including Shadow Event Adjustment Details in Invoices" in *BRM Designing and Generating Invoices*.	CM	Cached by the CM. Restart the CM after changing this entry.
subject	**pin_inv_send**	Specifies the subject of the e-mail. See "About Invoices" and "pin_inv_send" in *BRM Designing and Generating Invoices*.	**pin_inv**	The new value is effective immediately. You do not need to restart the CM.

Payments pin.conf Entries

Table 52-9 lists the Payments **pin.conf** entries.

Table 52-9 Payments pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
cc_collect	**fm_pymt_pol**	Specifies whether to perform real-time authorization of the fixed charges information with the customer's credit card during account creation. See "Charging Customers at Account Creation" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.
cc_revalidation_interval	**fm_pymt_pol**	Specifies the credit card revalidation interval.	CM	The new value is effective immediately. You do not need to restart the CM.
cc_validate	**fm_pymt_pol**	Specifies whether to validate a customer's credit card information during account creation. See "Validating Credit Cards at Account Creation" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.
cid_required	**fm_pymt_pol**	Specifies whether to use American Express CID (Card identifier) fraud protection for Paymentech transactions. See "Requiring Additional Protection against Credit Card Fraud" in *BRM Configuring and Collecting Payments*.	CM	The new value is effective immediately. You do not need to restart the CM.

Table 52-9 (Cont.) Payments pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
config_payment	**fm_pymt**	Specifies the database and POID of the **/config/payment** object. Change this value if you create a custom **/config/payment** object.	CM	The new value is effective immediately. You do not need to restart the CM.
cvv2_required	**fm_pymt_pol**	Specifies whether to use Visa CVV2 fraud protection for Paymentech transactions. See "Requiring Additional Protection against Credit Card Fraud" in *BRM Configuring and Collecting Payments*.	CM	The new value is effective immediately. You do not need to restart the CM.
dd_collect	**fm_pymt_pol**	Specifies whether to perform real-time authorization of the fixed charges information with the customer's debit card during account creation.	CM	The new value is effective immediately. You do not need to restart the CM.
dd_revalidation_interval	**fm_pymt_pol**	Specifies the debit card revalidation interval.	CM	The new value is effective immediately. You do not need to restart the CM.
dd_validate	**fm_pymt_pol**	Specifies whether to validate a customer's direct debit (debit card) information during account creation.	CM	The new value is effective immediately. You do not need to restart the CM.
merchant	**fm_cust_pol**	Specifies the merchant to receive money collected during credit-card processing. See "Specifying Merchant IDs and Merchant Numbers" in *BRM Configuring and Collecting Payments*.	CM	Cached by the CM. Restart the CM after changing this entry.
minimum	**pin_billd**	Specifies the minimum balance for retrieving accounts for collection. Use this entry to set a minimum threshold for the amount due on an account when searching for accounts for collection. See "Specifying the Minimum Payment to Collect" in *BRM Configuring and Running Billing*.	**pin_billd**	The new value becomes effective immediately. You do not need to restart the CM.
minimum_payment	**fm_pymt_pol**	Specifies the minimum amount to charge. The amount charged is greater than or equal to the minimum amount. See "Specifying the Minimum Amount to Charge" in *BRM Configuring and Running Billing*.	CM	The new value becomes effective immediately. You do not need to restart the CM.

Table 52-9 (Cont.) Payments pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
minimum_refund	**fm_pymt_pol**	Specifies the minimum refund amount that the system allows. The amount refunded is greater than or equal to the minimum amount. See "Specifying the Minimum Amount to Refund" in *BRM Managing Accounts Receivable*.	CM	The new value is effective immediately. You do not need to restart the CM.
payment_batch_lock	**fm_pymt**	Specifies whether Payment Tool locks accounts at the account level or the batch level when processing payments. See "Configuring Payment Tool to Lock at the Account Level during Batch Processing" in *BRM Configuring and Collecting Payments*.	CM	Cached by the CM. Restart the CM after changing this entry.
validate_acct	**fm_pymt_pol**	Allows use of the customer's credit card on an account different from the root account. See "Specifying the Account That Records Credit Card Validations" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.

Pricing and Rating pin.conf Entries

Table 52-10 lists the Pricing and Rating **pin.conf** entries.

Table 52-10 Pricing and Rating pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
backdate_trigger_auto_rerate	**fm_subs**	Specifies whether to create auto-rerate job objects used by **pin_rerate**.	CM	The new value is effective immediately. You do not need to restart the CM.
backdate_window	**fm_subs**	Specifies the minimum time difference needed between the current time and the backdated event time for triggering automatic rerating.	CM	The new value is effective immediately. You do not need to restart the CM.
cache_references_at_start	**fm_price**	Specifies whether to store objects referenced by price objects in memory in the CM cache when the CM starts. See "Improving Performance for Loading Large Product Offerings".	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-10 (Cont.) Pricing and Rating pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
cancel_tolerance	**fm_bill**	Specifies the cancellation tolerance of account charge offers in minutes. See "Canceling Charge Offers without Charging a Cancel Fee" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.
delay	**pin_rerate**	Specifies the delay for rerating jobs.	**pin_rerate**	The new value is effective immediately. You do not need to restart the CM.
extra_rate_flags	**fm_rate**	Turns optional rating features on and off.	CM	Cached by the CM. Restart the CM after changing this entry.
fm_offer_profile_cache	**cm_cache**	Specifies the attributes of the CM cache for offer profiles. **Note:** This entry is mandatory for policy-driven charging. See *BRM Creating Product Offerings*.	CM	Cached by the CM. Restart the CM after changing this entry.
fm_price_cache_beid	**cm_cache**	Specifies attributes of the CM cache for the size of balance element IDs (BEID). See "Improving Performance for Loading Large Product Offerings".	CM	The new value is effective immediately. You do not need to restart the CM.
fm_price_prod_provisioning_cache	**cm_cache**	Specifies the attributes of the CM cache for the size of charge offer provisioning tags. See "Improving Performance for Loading Large Product Offerings".	CM	The new value is effective immediately. You do not need to restart the CM.
log_price_change_event	**fm_price**	Specifies whether to create and log events for price object changes. See "Improving Performance for Loading Large Product Offerings".	CM	The new value is effective immediately. You do not need to restart the CM.
log_refresh_product	**fm_rate**	Specifies whether to log product offering changes. You can use the log entries to generate notifications about such changes. See "Logging Changes to Product Offerings" in *BRM Creating Product Offerings*.	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-10 (Cont.) Pricing and Rating pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
rate_change	**fm_subscription**	Enables the enhanced pricing change management feature. See "Rerating Cycle Forward and Cycle Forward Arrears Charges" in *BRM Rerating Events*.	CM	The new value is effective immediately. You do not need to restart the CM.
rating_max_scale	**fm_rate**	Specifies the precision level of decimal values. See "Changing the Precision of Rounded and Calculated Values".	CM	Cached by the CM. Restart the CM after changing this entry.
rating_quantity_rounding_scale	**fm_rate**	Specifies the precision of rounded values. See "Changing the Precision of Rounded and Calculated Values".	CM	Cached by the CM. Restart the CM after changing this entry.
rating_timezone	**fm_rate**	Specifies the server time zone.	CM	Cached by the CM. Restart the CM after changing this entry.
refresh_product_interval	**fm_rate**	Specifies the interval in which product offerings are refreshed in cache memory. See "Setting the Interval for Checking for Product Offering Changes".	CM	Cached by the CM. Restart the CM after changing this entry.
timezone_file	**fm_rate**	Specifies the location of the **timezones.txt** file.	CM	Cached by the CM. Restart the CM after changing this entry.
update_interval	**fm_zonemap_pol**	Specifies an interval for revising the value map. See "Setting the Interval for Checking for Product Offering Changes".	CM	Cached by the CM. Restart the CM after changing this entry.
validate_deal_dependencies	**fm_utils**	Specifies whether to validate the bundle prerequisites and mutually exclusive relations. See "Configuring Bundle Dependencies" in *BRM Managing Customers*.	CM	The new value is effective immediately. You do not need to restart the CM.

Revenue Assurance pin.conf Entries

Table 52-11 lists the Revenue Assurance **pin.conf** entries.

Table 52-11 Revenue Assurance pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
writeoff_control_point_id	**fm_process_audit**	Changes the control point ID. See *BRM Collecting Revenue Assurance Data*.	CM	Cached by the CM. Restart the CM after changing this entry.

Service Lifecycle Management pin.conf Entries

Table 52-12 lists the Service Lifecycle Management **pin.conf** entries.

Table 52-12 Service Lifecycle Management pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
fm_bill_template_cache	**cm_cache**	Specifies attributes of the **template** cache. This cache is used by the Service Lifecycle Management feature. See "Managing Service Life Cycles" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.
fm_bill_utils_business_profile_cache	**cm_cache**	Specifies attributes of the **business_profile** cache. This cache is used by the Service Lifecycle Management feature. See "Managing Service Life Cycles" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.
fm_cust_lifecycle_config_cache	**cm_cache**	Specifies attributes of the **lifecycle** cache. This cache is used by the Service Lifecycle Management feature. See "Managing Service Life Cycles" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.
fm_cust_statemap_config_cache	**cm_cache**	Specifies attributes of the **statemap** cache. This cache is used by the Service Lifecycle Management feature. See "Managing Service Life Cycles" in *BRM Managing Customers*.	CM	Cached by the CM. Restart the CM after changing this entry.

Services Framework pin.conf Entries

Table 52-13 lists the Services Framework **pin.conf** entries.

Table 52-13 Service Framework pin.conf Entries.

Name	Program	Description	pin.conf File	Implementation
agent_return	**fm_tcf**	Specifies the provisioning status when provisioning is simulated.	CM	Restart the CM after changing this entry.
commit_at_prep	**dm_provision**	Specifies whether to send the payload to the agent at PREP_COMMIT time and not to send anything at COMMIT time. See "Configuring the Provisioning Data Manager" in *BRM Telco Integration*.	**dm_prov_telco**	This entry is read by the utility when it runs. You do not need to restart the CM.
connect_retries	**dm_provision**	Specifies the number of times **dm_provision** attempts to connect to the Infranet agent on connection failure. See "Provisioning Data Manager Configuration File Entries" in *BRM Telco Integration*.	**dm_prov_telco**	This entry is read by the utility when it runs. You do not need to restart the CM.
connect_retry_interval	**dm_provision**	Specifies how many seconds to wait before retrying to connect to the Infranet agent on connection failure. See "Provisioning Data Manager Configuration File Entries" in *BRM Telco Integration*.	**dm_prov_telco**	This entry is read by the utility when it runs. You do not need to restart the CM.
prov_db	**fm_tcf**	Specifies the number of the database to which provisioning connects to send the service order. See "Connecting the Connection Manager to the Provisioning Data Manager" in *BRM Telco Integration*.	CM	Cached by the CM. Restart the CM after changing this entry.
prov_ptr	**dm_provision**	Specifies where to find the Provisioning Data Manager. See "Configuring the Provisioning Data Manager" in *BRM Telco Integration*.	CM	Cached by the CM. Restart the CM after changing this entry.
prov_timeout	**dm_provision**	Specifies the length of time to wait to receive a complete response from the provisioning agent. See the instructions in the **pin.conf** file.	**dm_prov_telco**	This entry is read by the utility when it runs. You do not need to restart the CM.
provisioning_enabled	**fm_tcf**	Enables charge offer provisioning.	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-13 (Cont.) Service Framework pin.conf Entries.

Name	Program	Description	pin.conf File	Implementation
simulate_agent	**fm_tcf**	Creates a response and updates the service order.	CM	The new value is effective immediately. You do not need to restart the CM.

Tax Calculation pin.conf Entries

Table 52-14 lists the Tax Calculation **pin.conf** entries.

Table 52-14 Tax Calculation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
commtax_series	**dm_vertex**	Specifies the Vertex tax package that BRM uses to calculate telecom taxes. See "Configuring the Vertex DM for Communications Tax Q Series" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.
commtax_config_path	**dm_vertex**	Specifies the location of Communications Tax Q Series configuration file (**ctqcfg.xml**). See "Configuring the Vertex DM for Communications Tax Q Series" in *BRM Calculating Taxes*.	**dm_vertex**	Valid only when **commtax_series** is set to **6**. Restart the DM after changing this entry.
commtax_config_name	**dm_vertex**	Specifies the Communications Tax Q Series configuration name. See "Configuring the Vertex DM for Communications Tax Q Series" in *BRM Calculating Taxes*.	**dm_vertex**	Valid only when **commtax_series** is set to **6**. Restart the DM after changing this entry.
dynamic_taxation	**fm_bill**	Specifies whether to disable dynamic taxation and to use the **taxcodes_map** file. This reverts tax calculation to how it was performed in BRM 12.0 Patch Set 4 and earlier releases. See "About Creating Tax Codes (Patch Set 4 and Earlier)" in *BRM Calculating Taxes*.	CM	Cached by the CM. Restart the CM after changing this entry.

Table 52-14 (Cont.) Tax Calculation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
include_zero_tax	**fm_rate**	Specifies whether to include zero tax amounts in the tax jurisdictions for the event's balance impacts. See "Reporting Zero Tax Amounts" in *BRM Calculating Taxes*.	CM	Cached by the CM. Restart the CM after changing this entry.
provider_loc	**fm_rate_pol**	Specifies the city, state, ZIP code, and country where services are provided for taxation.	CM	The new value becomes effective immediately and all subsequent tax calculations use the new address. You do not need to restart the CM.
quantum_logfile	**dm_vertex**	Specifies the QSUT API to log debug information (into the specified file name) about the current transaction to be processed for tax calculation. See "Setting Up Tax Calculation for Vertex" in *BRM Calculating Taxes*.	**dm_vertex**	The new value becomes effective immediately. You do not need to restart the CM.
quantumdb_passwd	**dm_vertex**	Specifies the Oracle user password. See "Configuring Vertex Manager" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.
quantumdb_register	**dm_vertex**	Specifies whether to log an audit trail of invoices in the Quantum Register database. See "Setting Up Tax Calculation for Vertex" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.
quantumdb_server	**dm_vertex**	Specifies the name of the database server than contains the Quantum tables. See "Configuring Vertex Manager" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.
quantumdb_source	**dm_vertex**	Specifies the schema where the STQ tables reside. For Indexed Sequential Access Method (ISAM) databases, specifies the ISAM data file directory. See "Configuring Vertex Manager" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.

Table 52-14 (Cont.) Tax Calculation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
quantumdb_user	**dm_vertex**	Specifies the name of the Oracle user. See "Configuring Vertex Manager" in *BRM Calculating Taxes*.	**dm_vertex**	Restart the DM after changing this entry.
tax_now	**fm_ar**	Specifies when tax is calculated for account-level adjustments (at the end of billing or at the time of adjustment).	CM	The new value is effective immediately. You do not need to restart the CM.
tax_return_juris	**fm_rate**	Specifies whether to summarize taxes by jurisdiction or to itemize taxes.	CM	Cached by the CM. Restart the CM after changing this entry.
tax_return_loglevel	**fm_rate**	Specifies how to log messages returned from the taxation DM.	CM	Cached by the CM. Restart the CM after changing this entry.
tax_reversal_with_tax	**fm_ar**	Specifies whether to apply a tax reversal for an adjustment, dispute, or settlement	CM	The new value is effective immediately. You do not need to restart the CM.
tax_supplier_map	**fm_rate**	Specifies the location of the **tax_supplier_map** file.	CM Billing utilities	Restart the CM after changing this entry.
tax_valid	**fm_cust_pol**	Specifies how to validate the state and zip code of the billing address during account creation.	CM	The new value is effective immediately. You do not need to restart the CM.
taxation_switch	**fm_bill**	Enables taxation. See "Enabling and Disabling Taxation Globally" in *BRM Calculating Taxes*.	CM	Cached by the CM. Restart the CM after changing this entry.
taxcodes_map	**fm_rate**	Specifies the location of the **taxcodes_map** file.	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
vertex_db	**fm_rate**	Specifies the database number of the Vertex database. See "Specifying Connection Entries" in *BRM Calculating Taxes*.	CM	The new value is effective immediately. You do not need to restart the CM.

53

System Administration pin.conf Reference

Learn how system administrators use **pin.conf** settings to improve the performance of Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Connection Manager (CM) pin.conf Entries

Table 53-1 lists the CM **pin.conf** entries.

Table 53-1 CM pin.conf Entries

Name	Program	Description
cm_cache_space	**cm**	(Optional) Reserves the cache memory for all the Facilities Modules. The default is **6291456**. The value is always in multiples of 1024.
cm_data_dictionary_cache	**cm_cache**	Increases the size of the CM cache for the data dictionary. See "Increasing the Size of the CM Cache for the Data Dictionary" in *BRM Developer's Guide*.
cm_data_file	**cm**	Specifies the name and location of the shared memory file that caches global information for the CM. See the instructions in the CM **pin.conf** file.
fm_offer_profile_cache	**cm_cache**	Loads all **offer_profile** objects at initialization of **fm_offer_profile** library.
cm_logfile	**cm**	Specifies the full path to the log file used by the CM.
cm_logformat	**cm**	Specifies which **PINLOG** format to use. See "Setting the CM Log Time Resolution".
cm_max_connects	**cm**	Specifies the maximum number of client connections to the CM. See "Specifying the Number of Connections to CMs".
cm_name	**cm**	Specifies the name of the computer where the CM runs. See "About Connecting BRM Components".
cm_opcode_stats	**cm**	Specifies whether to collect statistics about CM opcode performance. See "Getting Quality of Service Statistics".

Table 53-1 (Cont.) CM pin.conf Entries

Name	Program	Description
cm_ports	**cm**	Specifies the port number of the computer where the CM runs. See "About Connecting BRM Components".
cm_timeout	**cm**	Specifies the time-out value for receiving the next opcode request from an application. See "Setting the CM Time Interval between Opcode Requests".
creation_logging	**fm_cust**	Determines whether to log nondollar events. See "Logging Noncurrency Events" and "Improving Connection Manager Performance".
dm_attributes	**cm**	Specifies attributes associated with a particular database.
dm_pointer	**cm**	Specifies where to find one or more DMs for the BRM database. See "Setting Data Manager Attributes".
dm_port	**dm**	Specifies the port number of the computer where the CM runs. See "About Connecting BRM Components".
enable_pcm_op_call_stack	**dm**	Specifies whether PCM_OP_CALL_STACK is printed in the **cm.pinlog** file after each opcode from a **nap** (Network Application Program) is completed. See "Recording Opcode Calls in the CM Log File".
enable_preferences_enrichment	**fm_publish**	Specifies whether to enrich business events with your customers' preferences before they are sent to an external notification application. See "Enabling Notification Enrichment" in *BRM Managing Customers*.
fetch_size	**pin_billd** **pin_mta**	Specifies the number of records to retrieve from the database in a batch and cache in system memory for processing by the utility. See "Tuning the Account Cache Size for Billing Utilities (fetch_size)" and "Tuning the Account Cache Size for Invoice Utilities (fetch_size)".
group_children_fetch	**fm_bill**	Used to perform a step search for child accounts in groups when the parent group contains many members. See "Filtering Search Results".
group_members_fetch	**fm_bill**	Used to perform a step search for members of the group sharing object when the parent group contains many members. See "Filtering Search Results".
ip	N/A	Specifies the IP address of the devices managed by IP Address Manager.
item_fetch_size	**fm_bill**	Used to perform a step search for items. See "Filtering Search Results".
login_audit	**cm**	Creates the session event's object for each client application. See "Turning Off Session-Event Logging".

Table 53-1 (Cont.) CM pin.conf Entries

Name	Program	Description
max_pcm_op_call_stack_entries	**dm**	Specifies the number of entries allocated for PCM_OP_CALL_STACK. See "Recording Opcode Calls in the CM Log File".
monitor	**pin_mta**	Specifies the path and name of a shared memory map file used by the **pin_mta_monitor** utility.
passwd_age	**cm**	Specifies the number of days after which the password expires. See "Setting the Default Password Expiry Duration".
prefs_enabled_publisher_list	**fm_publish**	Specifies whether to enrich business events with your customers' preferences before they are sent to specific publishers. See "Enabling Notification Enrichment" in *BRM Managing Customers*.
prefs_phone_no_location	**fm_publish**	Specifies where BRM can find a customer's MSIDN number. See "Enabling Notification Enrichment" in *BRM Managing Customers*.
primary_database	N/A	Specifies the primary database schema. See the instructions in the **pin.conf** file.
primary_db	**cm**	Specifies the primary database schema. See "Running pin_multidb.pl -i" in *BRM Installation Guide*.
rps_no_reorder	**fm_price**	Specifies whether to reorder rate plan selectors before committing them to the database. • **1**: Leaves rate plan selectors in the order in which they appear in Pricing Center or the PCM_OP_PRICE_SET_PRICE_LIST opcode. • **0**: Specifies to reorder rate plan selectors before committing them to the database. This increases performance and reduces the buffer size. This is the default.
sample_handler_logfile	N/A	Specifies the full path to the log file for the Batch Controller. See "Configuring the Batch Controller" and "Customizing SampleHandler".
support_multiple_so	**fm_tcf**	Specifies whether Services Framework provisioning can make in-flight changes to service orders.

Data Manager (DM) pin.conf Entries

Table 53-2 lists the DM **pin.conf** entries.

Table 53-2 DM pin.conf Entries

Name	Program	Description
config_dir	**fm_cust_pol**	Specifies the location of the ISP configuration data. Used by the PCM_OP_CUST_POL_GET_CONFIG policy opcode. See "Sending Account Information to Your Application When an Account Is Created" in *BRM Managing Customers*.
crypt	NA	Associates a four-byte tag with an encryption algorithm and secret key combination. See "Configuring the Data Manager for Oracle ZT PKI Encryption" in *BRM Developer's Guide*.
dd_write_enable_fields	**dm**	Specifies whether this DM can create fields in the data dictionary. See "Modifying the pin.conf file to Enable Changes" in *BRM Developer's Guide*.
dd_write_enable_objects	**dm**	Specifies whether this DM can edit, create, and delete custom storable classes in the data dictionary. See "Modifying the pin.conf file to Enable Changes" in *BRM Developer's Guide*.
dd_write_enable_portal_objects	**dm**	Specifies whether this DM can delete predefined BRM storable classes and add and delete fields in one of those classes. See "Modifying the pin.conf file to Enable Changes" in *BRM Developer's Guide*.
dd_write_enable_types	**dm**	Specifies whether this DM can edit, create, and delete custom object type names in the data dictionary.
dm_bigsize	**dm**	Specifies the size of the DM shared memory. See "Setting DM Shared Memory Size".
dm_debug	**dm**	Specifies the debugging information to send to the log file. See the instructions in the **pin.conf** file.
dm_in_batch_size	**dm**	Specifies the number of objects to retrieve from subtables (arrays or substructs) in a search query. See the instructions in the **pin.conf** file.
dm_init_be_timeout	**dm**	Specifies the amount of time, in seconds, for the first DM backend to start before the DM times out. See "Setting How Long the DM Waits for the Background Startup Process to Complete".
dm_logfile	**dm**	Specifies the full path to the log file used by the CM.
dm_max_per_fe	**dm**	Specifies the maximum number of connections for each front end. See the instructions in the **pin.conf** file.
dm_mr_enable	**dm**	Specifies to use database cursors to fetch multiple rows. See "Configuring Oracle Databases" in *BRM Installation Guides*.
dm_n_be	**dm**	Specifies the number of back ends the program creates and uses. See "Configuring DM Front Ends and Back Ends".

Table 53-2 (Cont.) DM pin.conf Entries

Name	Program	Description
dm_n_fe	**dm**	Specifies the number of front ends the program creates and uses. See "Configuring DM Front Ends and Back Ends".
dm_port	**dm**	Specifies the port number for this DM.
dm_restart_children	**dm**	Specifies whether to replace child processes. See "Replacing Failed DM Child Processes".
dm_restart_delay	**dm**	Specifies the interval delay when DM back ends are spawned and respawned. See "Customizing BRM Multithreaded Client Applications" and "Configuring Your Multithreaded Application" in *BRM Developer's Guide*.
dm_sequence_cache_size	**dm**	(Optional) Specifies the number of Portal object IDs (POIDs) to be cached when each instance of a DM is started. The default is **1000**. The POIDs are cached in the memory when the DM is started. Whenever a database object is created, the DM uses the POIDs instead of accessing the database each time.
dm_shmsize	**dm**	Specifies the size of the shared-memory segment shared between the front and back ends for this BRM process. See "Setting DM Shared Memory Size".
dm_sql_retry	**dm**	Specifies the number of times an SQL statement is retried if the **ORA-01502: index 'PINPAP.I_EVENT_ITEM_OBJ__ID' or partition of such index is in unusable state** error occurs. **Note:** This is not a mandatory parameter in the **pin.conf** file. The default behavior is not to try running the SQL statement if the error occurs. See "Problem: ORA-01502: Index 'PINPAP.I_EVENT_ITEM_OBJ__ID' or Partition of Such Index Is in Unusable State".
dm_trans_be_max	**dm**	Specifies the maximum number of back ends that can be used for processing transactions. See "Configuring DM Front Ends and Back Ends".
dm_trans_timeout	**dm**	Specifies the time-out value for receiving the next opcode request from an application. See "Setting the DM Time Interval between Opcode Requests".
dm_xa_trans_timeout_in_secs	**dm**	Specifies how long, in seconds, an extended architecture (XA) transaction remains in an active state. By default, successfully prepared XA transactions expire in BRM if a commit request is not received within one hour after the transaction opens. This timeout is used only if the application server XA transaction timeout is not specified. The minimum value is **10**, and the maximum value is **5184000** (60 days). The default is **3600** (1 hour). See "Changing the XA Transaction Timeout Period" in *BRM JCA Resource Adapter*.

Table 53-2 (Cont.) DM pin.conf Entries

Name	Program	Description
extra_search	**dm**	Specifies whether to perform an extra search count (*) on subtables for optimal memory allocation. See "Optimizing Memory Allocation during Database Searches".
sm_database	**dm**	Specifies the database alias name. See "Connecting a Data Manager to the BRM Database".
sm_id	**dm**	Specifies the database user name that the DM uses to log in to the BRM database. This entry is set when you install BRM, but it can be changed. See "Connecting a Data Manager to the BRM Database".
sm_oracle_ddl	**dm**	Specifies whether to use Data Definition Language (DDL) when object types are updated in the data dictionary tables. See "Using DDL When Updating the Data Dictionary Tables" in *BRM Developer's Guide*.
sm_pw	**dm**	Specifies the password for the user specified in the **sm_id** entry. See "Configuring the Data Manager for Oracle ZT PKI Encryption" in *BRM Developer's Guide*.
stmt_cache_entries	**dm**	Specifies the maximum number of Oracle statement handles to cache to improve performance. See "How to Use the Statement-Handle Cache".

EAI Manager pin.conf Entries

Table 53-3 lists the EAI Manager **pin.conf** entries.

Table 53-3 EAI Manager pin.conf Entries

Name	Program	Description
dm_http_100_continue	**dm**	Specifies whether the DM waits for and reads a 100 Continue response. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.
dm_http_agent_ip	**dm**	Specifies a pointer to the HTTP agent. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.
dm_http_delim_crlf	**dm**	Specifies the HTTP server dependent delimiter used in the header. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.
dm_http_header_send_host_name	**dm**	Specifies that DM will send the host name as part of the header. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.

Table 53-3 (Cont.) EAI Manager pin.conf Entries

Name	Program	Description
dm_http_read_success	**dm**	Specifies whether the DM waits for and reads a 20x success response. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.
dm_http_url	**dm**	(Optional) Specifies complete URL of the HTTP server. See "Configuring EAI Manager to Publish to an HTTP Port" in *BRM Developer's Guide*.
eai_pointer	**cm**	Specifies where to find the EM that provides the opcode for the EAI Manager. See "Configuring the EAI DM" in *BRM Developer's Guide*.
em_eai_group	**cm**	Specifies the member opcode in a group provided by the EAI Manager.
em_group	**cm**	Specifies a member opcode in a group of opcodes provided by an EM. See "Configuring the Connection Manager for EAI" in *BRM Developer's Guide*.
enable_publish	**fm_publish**	Enables publishing of business events by using the EAI Manager. See "Configuring the Connection Manager for EAI" in *BRM Developer's Guide*.
plugin_name	**dm**	Specifies a pointer to a shared library that contains the code that implements the required interfaces of **dm_eai** as defined in **dm_eai_plugin.h**. See "Configuring the EAI DM" in *BRM Developer's Guide*.

Multithreaded Application (MTA) Framework pin.conf Entries

Table 53-4 lists the MTA Framework **pin.conf** entries.

Table 53-4 MTA Framework pin.conf Entries

Name	Program	Description
children	**pin_inv_accts** **pin_inv_export** **pin_inv_send** **pin_inv_upgrade** **pin_mta**	Specifies the number of worker threads spawned to perform the specified work. See "Tuning the Number of Children for Billing Utilities" and "Setting the Number of Children for Invoice Utilities".
cm_login_module	**cm**	Specifies the protocol for verifying applications that try to log in to the CM. See "Turning Off the Checking of Logons and Passwords".

Table 53-4 (Cont.) MTA Framework pin.conf Entries

Name	Program	Description
enable_ara	**pin_mta**	Enables revenue assurance activities through the various MTA applications (for example, **pin_bill_accts**, **pin_cycle_fees**, **pin_collect**, and **pin_inv_accts**). See "Configuring BRM Billing to Collect Revenue Assurance Data" in *BRM Collecting Revenue Assurance Data*.
enable_system_ara	**pin_mta**	Enables revenue assurance activities for your custom MTA application. This application can be run as part of a custom job in Business Operations Center. See "Enabling Your Custom Applications to Generate Revenue Assurance Data" in *BRM Collecting Revenue Assurance Data*.
fetch_size	**pin_mta**	Specifies the number of records to retrieve from the database in a batch and cache in system memory for processing by the utility. See "Tuning the Account Cache Size for Billing Utilities (fetch_size)".
hotlist	**pin_mta**	Specifies the path and file name of the hotlist file.
logfile	**pin_mta**	Specifies the name and location of the log file to record debug, warning, and error messages of running MTA-based applications. See "About the BRM MTA Framework" in *BRM Developer's Guide*.
login_name	**nap**	Specifies the user login name.
login_pw	**nap**	Specifies the login password.
login_type	**nap**	Specifies whether a login name and password are required.
loglevel	**pin_mta**	Specifies how much information is recorded in the log specified by the **logfile** parameter in the **pin.conf** file. See "Setting the Reporting Level for Logging Messages".
max_errs	**pin_mta**	Specifies the maximum number of errors allowed in the application.
max_time	**pin_mta**	Specifies the maximum time, measured from application start time, for job processing before the application exits.
multi_db	**pin_mta**	Specifies whether the application works with a BRM multischema system. See "Using Multithreaded Applications with Multiple Database Schemas" in *BRM Developer's Guide*.
per_batch	**pin_mta**	Specifies the number of account objects processed by each worker thread in batch mode.
per_step	**pin_mta**	Specifies how much data to store in **dm_oracle** when the utility is performing a step search. See "Setting the Batch Size for Invoice Utilities (per_step)".
respawn_threads	**pin_mta**	Specifies whether to respawn worker threads if they exit because of an error.
retry_mta_srch	**pin_mta**	Retries MTA searches when an insufficient memory error occurred.

Table 53-4 (Cont.) MTA Framework pin.conf Entries

Name	Program	Description
return_worker_error	**pin_mta**	Specifies whether to return an error code when there is an error in any of the threads.
userid	N/A	Specifies the CM for logging into the Data Manager.

54
business_params Reference

Learn how to use **business_params** entries to configure Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Accounts Receivable business_params Entries

Table 54-1 lists the Accounts Receivable **business_params** entries.

Table 54-1 Accounts Receivable business_params Entries

Name	Description	Implementation
AutoWriteOffReversal	Enables automatic write-off reversal on receipt of payment. See "Enabling Automatic Write-Off Reversals during Payment Collection" in *BRM Managing Accounts Receivable*.	Cached by the CM. Restart the CM after changing this entry.
CCRevalidationInterval	Specifies the time limit, in seconds, BRM waits before revalidating a customer's credit card during registration. BRM will not attempt to validate an account if a previous validation failed and the specified time has not elapsed. The default is **3600**.	Cached by the CM. Restart the CM after changing this entry.
Cvv2Required	For Paymentech credit-card processor users, specifies whether to require credit-card verification (CVV) data for Visa card transactions as a method of fraud prevention. The default is **disabled**.	Cached by the CM. Restart the CM after changing this entry.

Table 54-1 (Cont.) Accounts Receivable business_params Entries

Name	Description	Implementation
DDcollect	Specifies whether to collect a customer's current balance during registration. The default is **enabled**.	Cached by the CM. Restart the CM after changing this entry.
DDRevalidationInterval	Specifies the time limit, in seconds, BRM waits before revalidating a customer's direct debit account during registration. BRM will not attempt to validate an account if a previous validation failed and the specified time has not elapsed. The default is **3600** (1 hour).	Cached by the CM. Restart the CM after changing this entry.
PaymentIncentive	Enables payment incentives on early payment-in-full. See "Enabling BRM for Payment Incentives" in *BRM Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.
PaymentSuspense	Enables Payment Suspense Manager, which suspends payments exhibiting certain problems instead of failing or wrongly allocating them, and postpones them for later investigation. This enables the payment posting process to be completed without requiring immediate intervention to fix the errors. See "Enabling Payment Suspense in BRM" in *BRM Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-1 (Cont.) Accounts Receivable business_params Entries

Name	Description	Implementation
PerfAdvancedTuningSettings	Improves billing performance. The following values can be set for the **PerfAdvancedTuningSettings** business parameter. **0** (No Bit is set): Billing will be processed with default settings. **1** (0x01 Bit 0 is set): If this bit is set, the billing process will not set the item number on the bill items. Use this setting only if your system does not require items to have an item number. Item numbers appear in invoices, detailed ledger reports, and so on. **2** (0x02 Bit 1 is set): If this bit is set, the billing process will skip updating transfer events associated with the bill items. Use this setting only if your system does not have item transfer events created through line transfer operations. **4** (0x04 Bit 2 is set): Used internally by BRM **8** (0x08 Bit 3 is set): If this bit is set, the billing process will skip calculating the previous total unpaid bill for an open item accounting type bill unit. Use this setting only if your system does not require the previous unpaid amount reflected in the bill or the bill unit. See "Improving Performance by Skipping Billing-Time Tax Calculation". **16** (0x10 Bit 4 is set): If this bit is set, balance groups of the subordinate bill units will not be locked when the A/R parent bill unit is billed. For a large hierarchy, locking subordinate bill units can sometimes slow down other concurrent operations on the subordinate bill units. **32** (0x20 Bit 5 is set): When partial billing of an A/R parent bill unit is triggered, the partial billing of its subordinate bill units are triggered by default. If this bit is set, the partial billing of the subordinates will not be triggered before the A/R parent billing. **64** (0x40 Bit 6 is set): If this bit is set, the general ledger (G/L) collection is enabled for trial billing. When the G/L collection is enabled, the G/L **/journal** objects are created during trial billing. See "Improving Performance by Using Multiple Item Configurations". **Note:** By default, the value of this business parameter is 0. To set a value you can use a combination of values. For example, to skip updating transfer events and also skip calculating the previous total unpaid bill, set the value to 10 (2 + 8). **Important:** Do not set this business parameter to any value other than the valid values or any combination of those.	Cached by the CM. Restart the CM after changing this entry.
PINlessDebitProcessing	Enables BRM to process PINless debit payments. See "Enabling PINless Debit Payments in BRM" in *BRM Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-1 (Cont.) Accounts Receivable business_params Entries

Name	Description	Implementation
SearchBillAmount	Enables searches for **/bill** objects by total due amount.	Cached by the CM. Restart the CM after changing this entry.
WriteOffLevel	Specifies the level of write-off (account level, bill-unit level, or bill level) to track write-off reversals.	Cached by the CM. Restart the CM after changing this entry.

Billing business_params Entries

Table 54-2 lists the Billing **business_params** entries.

Table 54-2 Billing business_params Entries

Name	Description	Implementation
AcctCycleDelayPeriod	Use when billing occurred after an event was rated but before processing by RE Loader (that is, the event has not impacted the item). In such cases, RE Loader tries to locate the item from the next cycle if the event is created after billing in more than *X* days. See "Configuring an Accounting Cycle Delay Period" in *BRM Configuring and Running Billing*.	Not cached by the CM.
AllowCorrectivePaidBills	Determines whether to allow a corrective bill for a bill that is fully or partially paid. See "Enabling BRM to Create Corrective Bills for Partially or Fully Paid Bills" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
ApplyCycleFeeForBillNow	Indicates whether to apply cycle forward arrears and cycle arrears fees for Bill Now. See "Prorating Cycle Arrears and Cycle Forward Arrears for Bill Now" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
ApplyFolds	Specifies whether to apply folds and enable fold calculation. The default is **enabled**.	Cached by the CM. Restart the CM after changing this entry.
ApplyRollover	Specifies whether to apply rollovers. The default is **enabled**.	Cached by the CM. Restart the CM after changing this entry.
AutoTriggeringLimit	Suppresses auto-triggered billing for events processed after the **ConfigBillingDelay** interval. See "Configuring Auto-Triggered Billing" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
BillingCycleOffset	Specifies the hours of the day when the accounting and billing cycles start. See "Configuring the Billing Cutoff Time" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-2 (Cont.) Billing business_params Entries

Name	Description	Implementation
BillDomChargesOnCurrentBill	Specifies in which bill to include cycle forward charges when they occur on the billing day of month and within the billing delay period (**ConfigBillingDelay**): • **0**: The cycle forward charges are included in the next bill. This is the default. • **1**: The cycle forward charges are included in the current bill.	Cached by the CM. Restart the CM after changing this entry.
BillingFlowDiscount	Indicates the order of billing for discount-parents and discount-members (that is, who is billed first). See "Setting Up Billing for Charge and Discount Sharing Groups" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
BillingFlowSponsorship	Specifies the order of billing sponsors and sponsored accounts. See "Setting Up Billing for Charge and Discount Sharing Groups" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
BillTimeDiscountWhen	Enables billing-time discounts at the end of the billing cycle. See "Defining When Billing-Time Discounts Are Applied" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
CancelTolerance	Specifies the cancellation tolerance for account products, in minutes. This is the time after a product is purchased when it can be canceled with a full refund. For example, this tolerance is needed when a CSR assigns the wrong product to a customer and needs to cancel it. The default is **15**.	Cached by the CM. Restart the CM after changing this entry.
ConfigBillingCycle	Specifies how long after the end of the billing cycle that new events are included in the previous month's bill. The default is **0**. If this value is greater than **ConfigBillingDelay**, the system generates an error. See "Billing Cycle Override for Delayed Billing" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
ConfigBillingDelay	Specifies the number of days to wait beyond the billing day of month before billing subscribers. See "Configuring Delayed Billing" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
CorrectiveBillThreshold	Sets the appropriate threshold as the default value for creating corrective bills. See "Specifying the Minimum Threshold Amount for Corrective Bills" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
CreateTwoBillNowBillsInDelay	Creates two **Bill_Now** objects during the billing delay interval. The first **Bill_Now** object includes charges for the previous cycle. The second **Bill_Now** object contains charges from the events for the new (next) cycle.	Cached by the CM. Restart the CM after changing this entry.

Table 54-2 (Cont.) Billing business_params Entries

Name	Description	Implementation
CycleDelayAlign	Aligns the product purchase, cycle, and usage start and end times to the accounting cycle. This entry is applicable only when you configure delayed purchase, cycle, or usage start and end times when you set up your offers, and the delay is a whole number measured in cycles. See "Aligning Account and Cycle Start and End Times" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
CycleTaxInterval	Specifies how BRM calculates deferred taxes for wholesale billing. When set to **billing**, the tax is forwarded from the child account to the parent account. BRM calculates taxes for the parent account only, but the single tax item on the parent account includes taxes from both the parent and child accounts.	Cached by the CM. Restart the CM after changing this entry.
DealPurchaseForClosedAccount	Enables charge offer purchases from closed or inactive accounts. See "Enabling Charge Offer Purchases from Closed or Inactive Accounts" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
DeferredTaxJournaling	Specifies how BRM journals tax data for deferred tax calculation: • **0**: Skips journaling of tax data for deferred tax calculation. • **1**: Journals the tax data, but uses the default tax supplier (from configuration) and default tax locales (the subscriber's address) for tax calculation. It ignores any setting in the event being processed. It can also be used to indicate that the tax supplier and tax locales do not matter for the tax calculation. • **2**: Journals all tax data, including the tax supplier and tax locales. This is the default. **Note**: This parameter works independently from the **GeneralLedgerReporting** business parameter.	Cached by the CM. Restart the CM after changing this entry.
31DayBilling	Enables the 31-day billing feature. See "Configuring 31-Day Billing" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
EnableARA	Enables Revenue Assurance Manager. See "Enabling Billing Operations to Generate Revenue Assurance Data" in *BRM Collecting Revenue Assurance Data*.	Cached by the CM. Restart the CM after changing this entry.
EnableCorrectiveInvoices	Enables corrective billing and corrective invoicing. See "Enabling Corrective Billing" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
ExcludePercentForZeroTax	Specifies whether BRM reports a zero tax percentage amount. See "Reporting Taxes of Zero Percent" in *BRM Calculating Taxes*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-2 (Cont.) Billing business_params Entries

Name	Description	Implementation
GenerateCorrectiveBillNo	Determines the prefix and sequence number to assign to the corrective bill. See "Configuring Bill Numbers for Corrective Bills" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
GenerateJournalEpsilon	Indicates whether to apply rounding to item totals before calculating the bill total. See "Configuring BRM to Record Rounding Difference" in *BRM Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.
ItemEventChargeDiscountMode	Defines the search mode for the PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT opcode. The search criteria must match one of the following: • **0**: Any balance impact for the event. This is the default. • **1**: The gross charge of the balance impacts. • **2**: The net charge of the balance impacts. • **3**: The total charge of the balance impacts.	Cached by the CM. Restart the CM after changing this entry.
ItemizedTaxCalculation	Specifies whether to retrieve the total tax and tax due at each item level. The default is **disabled**.	Cached by the CM. Restart the CM after changing this entry.
MoveDayForward	Specifies an appropriate option for the 31 billing feature. If this feature is not used, the billing day of month (DOM) cannot be greater than 28. Otherwise, any day of the month can be used for billing. See "Setting the Forward and Back Billing Options" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
NonCurrencyResourceJournaling	Controls the creation of **/journal** objects for noncurrency balance impacts. See *BRM Collecting General Ledger Data*.	Cached by the CM. Restart the CM after changing this entry.
NonrefundableCreditItems	Specifies the types of items that will not be refunded with an outstanding credit balance. See "Defining Nonrefundable Items" in *BRM Managing Accounts Receivable*.	Cached by the CM. Restart the CM after changing this entry.
PaymentIncentive	Enables payment (grant) incentives when a payment is received early in the billing cycle. See "Enabling BRM for Payment Incentives" in *BRM Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.
ProdEndOffsetPlanTransition	Enables a phased-out service to remain active for any number of days between 1 and 31. See "Configuring Services for a Generation Change" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
RejectPaymentsForPreviousBill	Determines whether payments should be accepted or rejected when the bill number associated with a payment does not match the last bill.	Cached by the CM. Restart the CM after changing this entry.
RerateDuringBilling	Enables borrowing from the rollover amount during the current billing cycle.	Cached by the CM. Restart the CM after changing this entry.

Table 54-2 (Cont.) Billing business_params Entries

Name	Description	Implementation
RolloverCorrectionDuringBilling	Enables borrowing from the rollover amount during the current billing cycle.	Cached by the CM. Restart the CM after changing this entry.
SequentialCycleDiscounting	Enables BRM to evaluate cycle fee discounts purchased or canceled mid-cycle in conjunction with other discounts that are valid during the same period. See "Configuring Remaining Charge Cycle Discounting" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
ShortCycle	Specifies whether to create a short cycle. See "Specifying How to Handle Partial Accounting Cycles" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
SkipCheckForSubordinatesBilled	Enables you to skip validation of billing for nonpaying child bill units. See "Skipping Validation of Billing for Nonpaying Child Bill Units" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
SortValidityBy	Defines the default value for **consumption_rule**. When a customer uses a service, BRM must know which minutes (or sub-balance) to use first. You use consumption rules to specify the order in which sub-balances are consumed, according to the validity start time and end time. See "Setting the Default Consumption Rule" in *BRM Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.
SplitSponsorItemByMember	Enables you to divide accumulated charges across sponsored members of an account. See "Creating Custom Sponsored Bill Items" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
SSPRevenueRecognition	Enables you to use the deliverable-based revenue recognition scheme, which complies with the ASC 606 and IFRS 15 standards. See "Enabling Deliverable-Based Revenue Recognition" in *BRM Collecting General Ledger Data*.	Cached by the CM. Restart the CM after changing this entry.
StagedBillingFeeProcessing	Enables you to apply cycle forward fees in parallel. See "Applying Cycle Forward Fees in Parallel" in *PDC Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.
SubBalValidity	Allows you to extend the validity period of the original sub-balance when a subscription service is transferred. See "Configuring Sub-Balance Validity for Subscription Service Transfer" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
ValidateDiscountDependency	Enables discount exclusion rules, which establish a mutually exclusive relationship between discount offers or between a discount offer and a package. See "Enabling Mutually Exclusive Discount Offers" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-2 (Cont.) Billing business_params Entries

Name	Description	Implementation
WholesaleBillingSystem	Allows you to create only wholesale accounts and bill unit hierarchies. See "Setting Up Billing for Wholesale Account Hierarchies" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.

Customer business_params Entries

Table 54-3 lists the Customer **business_params** entries.

Table 54-3 Customer business_params Entries

Name	Description	Implementation
ActgDom	Specifies the default day of the month for billing accounts. The default is **0**.	Cached by the CM. Restart the CM after changing this entry.
BillWhen	Specifies the default number of accounting cycles before a customer is billed. The default is **1**.	Cached by the CM. Restart the CM after changing this entry.
CCchecksum	Specifies whether to run a checksum validation on the customer's credit card during registration. The default is **enabled**.	Cached by the CM. Restart the CM after changing this entry.
ClosedAcctsRetentionMonths	Specifies the number of months to retain deleted accounts in the BRM system. The default is **0**.	Cached by the CM. Restart the CM after changing this entry.
EnablePasswordRestriction	Enables passwords to secure the creation, modification, and deletion of **/service** objects. The default is **enabled**. See "Enabling Password Restriction for /service Objects".	Cached by the CM. Restart the CM after changing this entry.
SubscriberLifeCycle	Enables custom service life cycles. See "Enabling BRM to Use Custom Service Life Cycles" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.

General Ledger business_params Entries

Table 54-4 lists the General Ledger **business_params** entries.

Table 54-4 General Ledger business_params Entries

Name	Description	Implementation
CustomJournalUpdate	Enables custom updates to general ledger data. See "Customizing G/L Data Stored in /journal Objects" in *BRM Opcode Guide*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-4 (Cont.) General Ledger business_params Entries

Name	Description	Implementation
GeneralLedgerReporting	Specifies whether to enable or disable G/L collection and the creation of **/journal** objects. See "Disabling G/L Collection in BRM" in *BRM Collecting General Ledger Data*.	Cached by the CM. Restart the CM after changing this entry.
SegregateJournalsByGLPeriod	Records revenue in a separate journal entry for each G/L cycle in a billing cycle. See "Segregating Unbilled Revenue by G/L Cycle within a Billing Cycle" in *BRM Collecting General Ledger Data*.	Cached by the CM. Restart the CM after changing this entry.

Installment business_params Entries

Table 54-5 lists the Installment **business_params** entries.

Table 54-5 Installment business_params Entries

Name	Description	Implementation
DueDateReminderNotification	Specifies the number of days before an individual installment's due date that a reminder notification is generated. See "Configuring Installment Notification Days" in *Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.
EndDateReminderNotification	Specifies the number of days, before the effective date, when the last installment is due when the notification will be sent. See "Configuring Installment Notification Days" in *Configuring and Collecting Payments*.	Cached by the CM. Restart the CM after changing this entry.

Invoicing business_params Entries

Table 54-6 lists the Invoicing **business_params** entries.

Table 54-6 Invoicing business_params Entries

Name	Description	Implementation
ADSTTaxHandle	Groups taxes on invoices based on the tax supplier IDs. See "Aggregating Taxes on Invoices" in *BRM Designing and Generating Invoices*.	Cached by the CM. Restart the CM after changing this entry.
ARItemsInCorrectiveInvoice	Ensures that all adjustments are allocated to the corrective bill and displayed only in the corrective invoice whenever it is generated. See "About Deferred Corrective Invoices" in *BRM Designing and Generating Invoices*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-6 (Cont.) Invoicing business_params Entries

Name	Description	Implementation
EnableInvoicingIntegration	Generates invoice documents by using the BRM-BI Publisher integration framework. See "Enabling the BRM-BI Publisher Integration" in *BRM Designing and Generating Invoices*.	This entry is read by the utility when it runs. You do not need to restart the CM.
InvoiceStorageType	Specifies the format in which to store invoices in the database. See "Specifying the Default Format in Which to Store Invoices in BRM" in *BRM Designing and Generating Invoices*.	Cached by the CM. Restart the CM after changing this entry.
PromotionDetailDisplay	Displays promotion details on invoices. See "Specifying Whether BRM Displays Promotion Details on Invoices" in *BRM Designing and Generating Invoices*.	Cached by the CM. Restart the CM after changing this entry.
SubARItemsIncluded	Displays A/R items for each nonpaying bill unit on the paying parent invoice. See "Setting Defaults for Hierarchical Bill Unit Invoices" in *BRM Designing and Generating Invoices*.	This entry is read by the utility when it runs. You do not need to restart the CM.
ThresholdSubordsDetail	Sets the threshold for including nonpaying child bill unit details on parent detailed invoices. See "Setting Defaults for Hierarchical Bill Unit Invoices" in *BRM Designing and Generating Invoices*.	This entry is read by the utility when it runs. You do not need to restart the CM.
ThresholdSubordsSummary	Sets the threshold for including nonpaying child bill unit details on parent summary invoices. See "Setting Defaults for Hierarchical Bill Unit Invoices" in *BRM Designing and Generating Invoices*.	This entry is read by the utility when it runs. You do not need to restart the CM.

Multibalance business_params Entries

Table 54-7 lists the Multibalance **business_params** entries.

Table 54-7 Multibalance business_params Entries

Name	Description	Implementation
BalanceMonitoring	Specifies whether to monitor currency balances. The default is **disabled**. See "Enabling Balance Monitoring in BRM" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
ConsumeSubType	Specifies the order of loan or main balance consumption processing. The valid values are: • **main**: Main balance consumption processing. • **loan**: Loan balance consumption process. This is the default.	Cached by the CM. Restart the CM after changing this entry.

Table 54-7 (Cont.) Multibalance business_params Entries

Name	Description	Implementation
CreditThresholdChecking	Specifies whether to perform credit threshold checking during the batch rating process. The valid values are: • **enabledOffline**: Specifies to check credit thresholds during the batch rating process. • **disabled**: Specifies to skip credit threshold checking during the batch rating process and thus increase pipeline processing performance. This is the default. See "Enabling Threshold Checking in Pipeline Manager" in *BRM Configuring Pipeline Rating and Discounting*.	This entry is read by the utility when it runs. You do not need to restart the CM.
LockConcurrency	Indicates the concurrency of object locking. The valid values are: • **normal**: Specifies to lock the account object. • **high**: Specifies to have more concurrency of locking with greater granularity in terms of which balance group to lock. This is the default. See "Disabling Granular Object Locking" in *BRM Developer's Guide*.	Cached by the CM. Restart the CM after changing this entry.
RestrictResourceValidityToOffer	Restricts balance validity end time to the end time of the charge offer or discount offer that grants the balance. See "Restricting the End Time of Granted Balances That Start on First Usage" in *BRM Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.
SortValidityBy	Specifies the default consumption rule when consuming validity-based sub-balances. The default is **ESTEET**. See "Setting the Default Consumption Rule" in *BRM Creating Product Offerings*.	This entry is read by the utility when it runs. You do not need to restart the CM.

Notification business_params Entries

Table 54-8 lists the Notification **business_params** entries.

Table 54-8 Notification business_params Entries

Name	Description	Implementation
KafkaDBNumber	Specifies the Kafka database number. This value replaces the **{Dynamic}** key variable in message payloads sent to Kafka topics. See "Configuring the Dynamic Key Value" in *BRM Developer's Guide*.	Cached by the CM. Restart the CM after changing this entry.

Pricing and Rating business_params Entries

Table 54-9 lists the Pricing and Rating **business_params** entries.

Table 54-9 Pricing and Rating business_params Entries

Name	Description	Implementation
AllocateReratingAdjustments	Determines whether to allocate automatic adjustments from rerating to original bills. See "Creating Corrective Bills" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
AllowBackdateNoRerate	Enables backdating beyond the number of billing cycles specified in the **num_billing_cycles** entry without requesting to automatically rerate.	Cached by the CM. Restart the CM after changing this entry.
ApplyDeferredTaxDuringRerating	Enables deferred tax calculation during rerating. See "Enabling Deferred Tax Calculation during Rerating" in *BRM Rerating Events*.	Cached by the CM. Restart the CM after changing this entry.
ApplyDiscountOnZeroCharge	Specifies whether to apply discounts when the product scale is 0. When disabled, rating skips the pipeline call so that discounts are not processed if the product scale is 0. The default is **enabled**.	Cached by the CM. Restart the CM after changing this entry.
CreateRerateJobDuringCancel	Enables the automatic creation of rerate jobs when customers cancel products with noncurrency resources. See "Enabling Rerating for Canceled Noncurrency Resources" in *BRM Rerating Events*.	Cached by the CM. Restart the CM after changing this entry.
ECERating	Enables Oracle Communications Elastic Charging Engine (ECE) rating. See "Configuring the Connection Manager for Rerating" in *BRM Rerating Events*.	Cached by the CM. Restart the CM after changing this entry.
EnableEras	Specifies how to enable ERAs. See "Filtering the ERAs Considered during Rating and Discounting".	Cached by the CM. Restart the CM after changing this entry.
EnableGlobalChargeSharing	Enables global charge sharing. See "Enabling Global Charge Sharing Searches during Discounting" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
OfferEligibilitySelectionMode	Enables BRM to rerate events by using the pricing applied when rating conditions change during the session. See "Enabling Rerating when the Rating Conditions Change During the Session" in *BRM Rerating Events*.	Cached by the CM. Restart the CM after changing this entry.
ProductsDiscountsThreshold	Specifies the maximum number of charge offers or discount offers that can be cached for rating. See "Configuring the Maximum Number of Charge and Discount Offers Cached".	Cached by the CM. Restart the CM after changing this entry.
RatePreCacheProductAndDisc	Specifies whether system products, user products, system discounts, or user discounts are cached during CM startup. Bit fields of the integer value are used to specify the configuration options: • Bit 0: Pre-caches system products. • Bit 1: Pre-caches user products. • Bit 2: Pre-caches system discounts. • Bit 3: Pre-caches user discounts. The default is **0**.	Cached by the CM. Restart the CM after changing this entry.

Table 54-9 (Cont.) Pricing and Rating business_params Entries

Name	Description	Implementation
ResetMemberCreditLimit	Specifies whether a balance monitor group member's credit limit is set to NULL when its credit limit is rolled up to the parent. When enabled, a member's credit limit is set to NULL. The default is **enabled**. See "Nullifying Credit Limits for PR_RTCE Child Members" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
TimestampRoundingForPurchaseGrant	Enables time-stamp rounding to midnight for the resources granted by purchase events. See "Configuring Time-Stamp Rounding for Purchase Events" in *PDC Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.

Subscription business_params Entries

Table 54-10 lists the Subscription **business_params** entries.

Table 54-10 Subscription business_params Entries

Name	Description	Implementation
ApplyChargeOnInactiveOrCancelledProduct	Specifies whether to apply charges to inactive or canceled products. The default is disabled. See "Specifying Whether to Charge Inactive, Canceled, or SuspendedActive Accounts" in *PDC Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.
ApplyRolloverBeforeCycleFees	Specifies whether PCM_OP_SUBSCRIPTION_CYCLE_FORWARD adds rollover details to **/event/notification/subscription/renewal** notification events. See "Adding Rollover Details to Subscription Renewals" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
AutomatedGroupSharingSetup	Specifies whether to automatically create a discount sharing group or product sharing group for the top-level parent or billing account. The valid values are: • **0**: Disabled. This is the default. • **1**: Automatic discount sharing is enabled. • **2**: Product sharing groups are enabled. • **3**: Both automatic discount sharing and product sharing groups are enabled. See "Enabling AMS and Automated Sharing Groups in BRM" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
AutomatedMonitorSetup	Specifies whether to use Automated Monitor Setup (AMS) for creating balance monitor groups, product sharing groups, and discount sharing groups. See "Enabling AMS in BRM" or "Enabling AMS and Automated Sharing Groups in BRM" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-10 (Cont.) Subscription business_params Entries

Name	Description	Implementation
CancelFullDiscountImmediate	Enables BRM to cancel discounts immediately when the mid-cycle purchase/cancel option is set to **Full Discount**.	Cached by the CM. Restart the CM after changing this entry.
CancelledOfferingsSearch	Specifies whether to disable searches on closed offerings. See "Excluding Searches on Closed Offerings".	Cached by the CM. Restart the CM after changing this entry.
CreateTwoEventsInFirstCycle	Specifies how many events BRM uses when prorating a cycle fee for a charge offer whose validity expires within the first cycle after the charge offer was purchased. When this parameter is enabled, a charge event for the full cycle and a refund event for the unused portion are generated. When disabled (the default), only a charge event for the used portion of the cycle is generated. See "Using Two Events to Prorate Charges for Charge Offers Whose Validity Ends in First Cycle" in *BRM Configuring and Running Billing*.	Cached by the CM. Restart the CM after changing this entry.
DefaultZoneMapName	Specifies the name of the default zone map to use for location-based operations.	Cached by the CM. Restart the CM after changing this entry.
EventAdjustmentsDuringCancellation	Specifies whether to include event adjustments when calculating charge offer cancellation refunds. See "Including Event Adjustments in Charge Offer Cancelation Refunds" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
LoanRepaymentPercent	In cases where customers make an underpayment, specifies the percentage of the payment that is applied to the loan repayment. Valid values are 0 through 100, where 0 specifies that no amount is applied to the loan repayment and 100 specifies that the entire amount is applied to the loan repayment. The default is **100**.	Cached by the CM. Restart the CM after changing this entry.
MaxServicesToSearch	Enables you to specify the batch size of the number of services to search. See "Improving Performance in Retrieving Purchased Offerings for a Bill Unit".	Cached by the CM. Restart the CM after changing this entry.
ProductLevelValidation	Specifies whether to perform bundle dependency validations on inactive or canceled charge offers and discount offers. See "Configuring Bundle Dependencies" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
RolloverTransfer	Enables rollover transfers.	Cached by the CM. Restart the CM after changing this entry.
SubsDis74BackDateValidations	Enables users to create accounts with services or balances backdated prior to the account creation date. See "Allowing Accounts To Be Created with Backdated Services or Balances" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.

Table 54-10 (Cont.) Subscription business_params Entries

Name	Description	Implementation
TransferScheduledActions	Specifies whether to transfer pending scheduled actions associated with an existing subscription when transferring the subscription to a different account. See "About Transferring a Service Group to Another Account" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
UsePrioritySubscriptionFees	When multiple charge offers in a bundle include recurring charges, specifies whether to apply recurring charges in the order of charge offer priority. See "Applying Recurring Charges Based on Charge Offer Priority" in *PDC Creating Product Offerings*.	Cached by the CM. Restart the CM after changing this entry.

System Administration business_params Entries

Table 54-11 lists the System Administration **business_params** entry.

Table 54-11 Systems Administration business_params Entry

Name	Description	Implementation
AcceptableDelayTime	Specifies how long after the delivery time that messages can be delivered to your customers through an external notification application. The default is 2 hours. See "Setting Systemwide Values for Notifications" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
ConfigCacheRefreshInterval	Specifies the interval, in minutes, after which configuration parameters in the cache are considered stale. After the specified amount of time, the CM updates the configuration parameters in its cache. The default is **0**.	Cached by the CM. Restart the CM after changing this entry.
CrossSchemaSharingGroup	Specifies whether the parent and member accounts in sharing groups can reside in different database schemas. The default is **disabled**. See "Enabling Group Members to Reside in Multiple Schemas" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
EnableExternalIds	Specifies whether to generate unique IDs for **/event**, **/balance_group**, **/billinfo**, and **/payinfo** objects. A value of **1** enables the generation of unique IDs, and **0** disables it. The default is **0**.	Cached by the CM. Restart the CM after changing this entry.
LockConcurrency	Locks the account object, system-wide, at the account level or balance-group level. See "Disabling Granular Object Locking" in *BRM Developer's Guide*.	Cached by the CM. Restart the CM after changing this entry.
MaxLoginAttempts	Specifies the maximum number of invalid login attempts allowed with an incorrect password. See "Configuring the Maximum Number of Invalid Login Attempts".	Cached by the CM. Restart the CM after changing this entry.

Table 54-11 (Cont.) Systems Administration business_params Entry

Name	Description	Implementation
NotificationSilentPeriod	Specifies the silent period during which messages cannot be delivered to customers through an external notification application. The default is 21:00 through 07:00. See "Setting Systemwide Values for Notifications" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
NotificationSubscriberPreferences	Specifies the list of subscriber preferences to add to messages before they are sent to an external notification application. See "Adding Custom Fields during Enrichment" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.
PrepaidPartitionSet	Specifies the partition set for prepaid events. The valid values are 0, and 2 through 7. When set to **0**, prepaid partitioning is disabled. The default is **0**. See "Enabling Prepaid Event Partitions in BRM" in *ECE Implementing Charging*.	Cached by the CM. Restart the CM after changing this entry.
SilentDaysCalendarName	Specifies the name of the calendar for determining the silent days. Notifications cannot be delivered to your customers on silent days. The default is **NotificationSilentDays**. See "Setting Systemwide Values for Notifications" in *BRM Managing Customers*.	Cached by the CM. Restart the CM after changing this entry.

Part X
PDC System Administration

This part describes Oracle Communications Pricing Design Center (PDC) system administration tasks. It contains the following chapters:

- Administering Pricing Design Center
- Troubleshooting Pricing Design Center
- Monitoring PDC with Prometheus and Grafana

55

Administering Pricing Design Center

Learn about basic Oracle Communications Pricing Design Center (PDC) administration tasks.

Topics in this document:

Managing PDC Security

Oracle WebLogic Server includes a security architecture that provides a secure foundation for applications. PDC depends on the WebLogic Server security framework to secure its resources and servers. Managing PDC security involves the following:

- Managing security realms: Configuring new security realms, changing the default security realm, and deleting security realms.
- Managing users and groups: Defining users and assigning them to a group that can be authenticated in a security realm.
- Managing security providers: Managing security providers that provide security services to applications to protect WebLogic Server resources.
- Managing security policies: Managing security policies that specify who can access a WebLogic Server resource.

  The **Wssp1.2-2007-Https-BasicAuth.xml** security policy secures the connection between the PDC Web service API and the client applications using one way secure sockets layer (SSL). You can also add custom security policies to secure the PDC Web service API using WebLogic Server Administration Console.

See the WebLogic Server Administration Console Help for more information on managing security realms, managing users and groups, managing security providers, and managing security policies.

Monitoring PDC

Monitoring your system regularly ensures fast recognition and resolution of problems or issues. You can use WebLogic Server Administration Console to monitor the following:

- The PDC domain

- The PDC administration server and managed servers

See the WebLogic Server Administration Console Help for more information.

Managing PDC

Managing PDC involves managing the following:

- The PDC database
- The WebLogic server on which PDC is installed

Managing the PDC Database

To manage the PDC database and perform administrative tasks, see the Oracle Database Server documentation.

Managing the PDC WebLogic Server

Managing the WebLogic server on which PDC is installed involves:

- Starting and stopping PDC
- Starting and stopping the PDC administration server and managed servers

See the WebLogic Server Administration Console Help for more information on managing the WebLogic server on which PDC is installed.

Managing the PDC Transformation Process

Pricing and setup components configured in PDC must be published to Oracle Communications Billing and Revenue Management (BRM) subscription and rating engines, referred to as the target engines. During the publication process, the data is transformed into the format used by the BRM subscription and rating engines. The rating engines include BRM real-time rating engine (RRE), BRM batch-rating engine (BRE), and Oracle Communications Billing and Revenue Management (BRM) Elastic Charging Engine (ECE).

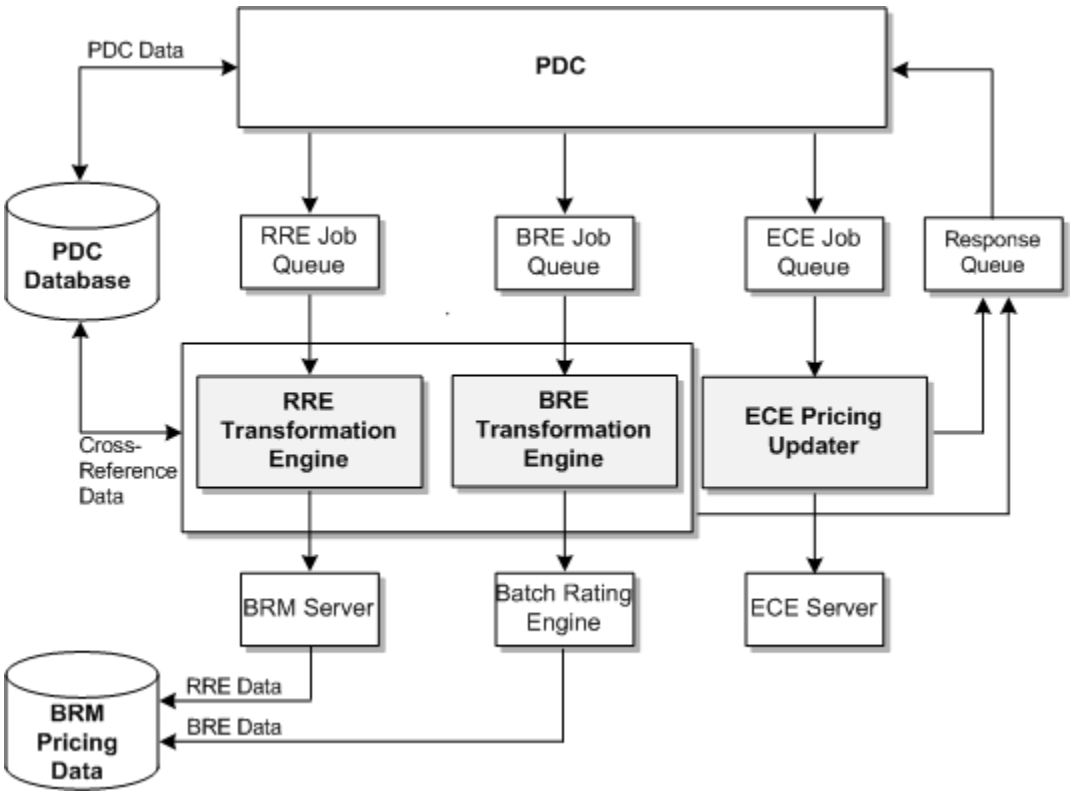
The PDC transformation process for pricing and setup components involves the following steps:

1. Using the PDC GUI application or the **ImportExportPricing** utility, you submit the pricing and setup components that you want to publish.
2. PDC validates the pricing and setup components.
3. PDC creates a transformation job for each target engine.
4. The job dispatcher puts the transformation job in the appropriate target engine work item queue.
5. The transformation engine or the ECE Pricing Updater associated with that queue retrieves the PDC components from the job dispatcher (see "About the Transformation Engines" and "About the ECE Pricing Updater").
6. The transformation engine or the ECE Pricing Updater converts the PDC components into the format required by the target engine (see "About the Target Engine Load Utilities" and "About the ECE Pricing Updater").

7. The transformation engine calls the load utility in the BRM server and Batch Rating Engine to load the PDC components into the respective databases, and ECE Pricing Updater loads the PDC components into ECE.
8. The transformation engine updates the tables in its cross-reference database.
9. The transformation engine or the ECE Pricing Updater sends the status to the job dispatcher, which notifies the PDC server.
10. The PDC server updates the status of the changeset.

Figure 55-1 illustrates the PDC transformation process for the ECE and BRM integration.

Figure 55-1 PDC Transformation Process



About the Transformation Engines

PDC uses transformation engines to convert pricing and setup components into the XML format required by the target engines.

The following transformation engines are included with PDC and are installed when you install the BRM Integration Pack:

- **RRETransformationEngine:** This transformation engine supports the BRM real-time rating engine.
- **BRETransformationEngine:** This transformation engine supports the BRM batch rating engine.

The transformation engines use information in the *BRM_Integration_Pack_home***/apps/transformation/TransformationConfiguration.xml** file (where

BRM_Integration_Pack_home is the directory in which you installed BRM Integration Pack) to connect to their target engine, to the cross-reference database, to the PDC server, and to the log files of the target engine.

About the Target Engine Load Utilities

After converting pricing and setup components into the appropriate XML format, the RRE and BRE transformation engines call their associated load utilities to load the components into the respective databases. The load utilities are as follows:

- **loadpricelist:** The RRE transformation engine uses this utility to load pricing and setup components into the target BRM database. To connect to the target database, this utility uses the **Infranet.properties** file. For more information, see the BRM documentation.
- **LoadIfwConfig:** The BRE transformation engine uses this utility to load pricing and setup components into the target BRM database. To connect to the target database, this utility uses the **LoadIfwConfig.reg** file. For more information, see the BRM documentation.

About the ECE Pricing Updater

The ECE Pricing Updater loads pricing and setup components into ECE automatically when you publish pricing and setup components in PDC for its target engine. The ECE Pricing Updater is packaged with ECE. For more information, see the ECE documentation.

Troubleshooting Transformation Errors

See the discussion about troubleshooting PDC in the *PDC Installation Guide* for information about troubleshooting transformation errors.

Starting the Transformation Engines

The transformation engines do not automatically start when you start the PDC server. You must manually start them by running the following scripts:

- For real-time rating: *BRM_Integration_Pack_home***/Transformation/apps/bin/startRRETransformer**
- For batch rating: *BRM_Integration_Pack_home***/Transformation/apps/bin/startBRETransformer**. This script is optional if you are using ECE for usage rating.

When you run the scripts, at the command prompt, enter the BRM Integration Pack wallet password and start the transformation engines.

Status of the Transformation Engines

If you suspect that a transformation engine has stopped running, use standard system commands to check the status of the transformation engine process.

Following are the names of the BRM transformation engine processes:

- For real-time rating: **oracle.communications.brm.pdc.server.transformation.rre.RRETransformationEngine**
- For batch rating: **oracle.communications.brm.pdc.server.transformation.bre.BRETransformationEngine**

Stopping the Transformation Engines

Use the following scripts to stop the transformation engines:

- For real-time rating: *BRM_Integration_Pack_home***/Transformation/apps/bin/stopRRETransformer**
- For batch rating: *BRM_Integration_Pack_home***/Transformation/apps/bin/stopBRETransformer**

Creating the Oracle Wallet

PDC stores the configuration entries and sensitive information, such as account and database passwords, in the PDC and BRM Integration Pack wallets. PDC provides sample PDC and BRM Integration Pack wallets. You can use these wallets to store the configuration data and passwords.

To create your own PDC wallet to store the configuration data and passwords, run the following command:

```
./PDCWalletUtil.sh create walletlocation walletpassword
```

where:

- *walletlocation* is the location in which the wallet must be created.
- *walletpassword* is the wallet password.

To create your own BRM Integration Pack wallet to store the configuration data and passwords for the utilities in BRM Integration Pack, run the following command:

```
./PDCBRMWalletUtil.sh create walletlocation walletpassword
```

> **Note:**
>
> Ensure that the following wallet files are created in the specified wallet location:
>
> - **cwallet.sso**
> - **cwallet.sso.lck**
> - **ewallet.p12**
> - **ewallet.p12.lck**

After you create the wallets, update the appropriate wallet location in the following configuration files:

- *PDC_home***/apps/bin/ImportExportConfiguration.xml.** Specify the PDC wallet location in this file. This file is used to configure the **ImportExportPricing** utility. See "Importing and Exporting Pricing and Setup Components" in *PDC Creating Product Offerings*.
- *BRM_Integration_Pack_home***/apps/syncpdc/SyncPDCConfiguration.xml.** Specify the BRM Integration Pack wallet location in this file. This file is used to configure the **SyncPDC** utility. See "Synchronizing Pricing Setup Components" in *PDC Creating Product Offerings*.
- *BRM_Integration_Pack_home*/**apps/transformation/TransformationConfiguration.xml.** Specify the BRM Integration Pack wallet location in this file. This file is used by the transformation engines to connect to their target engine.
- *BRM_Integration_Pack_home***/apps/migration/MigrationConfiguration.xml.** Specify the BRM Integration Pack wallet location in this file. This file is used to configure the **MigrateBRMPricing** utility. See "BRM Pricing Migration Utility" in *BRM Upgrade Guide*.

Changing Passwords in the Wallet

PDC stores the passwords for the WebLogic Server domain, PDC user, transformation cross-reference database, migration cross-reference database, and BRM database in the PDC and BRM Integration Pack wallets. To change the passwords in these wallets, see the following topics:

- Changing the Password in the PDC Wallet
- Changing the Password in the BRM Integration Pack Wallet

Changing the Password in the PDC Wallet

To change the password in the PDC wallet:

1. Go to the *PDC_home***/apps/bin** directory, where *PDC_home* is the directory in which the PDC software is installed.
2. Run the following command:

   ```
   ./PDCWalletUtil.sh walletlocation walletpassword configentry password
   ```

   where:

   - *walletlocation* is the location in which the PDC wallet must be created.
   - *walletpassword* is the PDC wallet password.
   - *configentry* is the configuration entry to store the password. The valid values are:
     - **PDC_APP_SERVER_USER_PASSWORD.** Specifies the user password for the PDC server.
     - **PDC_APP_USER_PASSWORD.** Specifies the PDC user password.
     - **TARGET_PDC_ADMIN_PASSWORD.** Specifies the password for the PDC administrator in the target PDC system.
     - **SOURCE_PDC_ADMIN_PASSWORD.** Specifies the password for the PDC administrator in the source PDC system.

  - **TARGET_PDC_USER_PASSWORD.** Specifies the password for the PDC user in the target PDC system.
  - **SOURCE_PDC_USER_PASSWORD.** Specifies the password for the PDC user in the source PDC system.
- *password* is the password to be stored.

3. At the command prompt, enter **Y**.

The password is changed in the PDC wallet.

Changing the Password in the BRM Integration Pack Wallet

To change the password in the BRM Integration Pack wallet:

1. Go to the *BRM_Integration_Pack_home***/apps/bin** directory.
2. Run the following command:

   ```
   ./PDCBRMWalletUtil.sh walletlocation walletpassword configentry password
   ```

   where:

   - *walletlocation* is the location in which the BRM Integration Pack wallet must be created.
   - *walletpassword* is the BRM Integration Pack wallet password.
   - *configentry* is the configuration entry to store the password. The valid values are:
     - **MIGRATION_DB_PASSWORD.** Specifies the migration cross-reference database password.
     - **TRANS_XREF_DB_PASSWORD.** Specifies the transformation cross-reference database password.
     - **BRM_DB_PASSWORD.** Specifies the BRM database password.
   - *password* is the password to be stored.
3. At the command prompt, enter **Y**.

The password is changed in the BRM Integration Pack wallet.

Changing the SQL and EclipseLink Log Level for PDC

To change the EclipseLink log level for PDC:

1. Navigate to the *MW_home***/oracle_common/common/bin** directory, where *MW_home* is the directory in which the Oracle Middleware components are installed.
2. Start the WebLogic Scripting Tool (WLST) by running the following command:

   ```
   ./wlst.sh
   ```

3. Connect to the server on which you want to change the logging level by running the following command:

   ```
   connect(UserName,Password,'t3://hostname:port')
   ```

4. Go to the custom settings by running the following command:

   ```
   custom()
   ```

> **Note:**
>
> **'custom()'** can take a few minutes to run, approximately 5 minutes.

5. Go to TopLink by running the following command:

   ```
   cd('TopLink')
   ```

6. List the sessions at this level by running the following command:

   ```
   ls()
   ```

   For example:

   ```
   drw-TopLink:Name=Development-JobDispatcher#1.2.0.0.0file_/scratch/ri-user-1/
   data/pdc_domain/servers/AdminServer/tmp/WL_user/
   JobDispatcher_1.2.0.0.0/7qjzil/APP-INF/lib/
   jobdispatcher_entities.jar_default,Type=Configuration
   drw-TopLink:Name=Development-pricingui#V2.0file/scratch/ri-user-1/data/
   pdc_domain/servers/AdminServer/tmp/_WL_user/pricinguiV2.0/jztz5c/APP-INF/lib/
   pricing_entities.jar_default,Type=Configuration
   drw-TopLink:Name=Session(JobDispatcher#1.2.0.0.0file/scratch/ri-user-1/data/
   pdcdomain/servers/AdminServer/tmp/_WL_user/JobDispatcher1.2.0.0.0/7qjzil/APP-
   INF/lib/jobdispatcher_entities.jar_default)
   drw-TopLink:Name=Session(pricingui#V2.0file_/scratch/ri-user-1/data/
   pdc_domain/ser vers/AdminServer/tmp/WL_user/pricinguiV2.0/jztz5c/APP-INF/lib/
   pricing_entiti es.jar_default)
   ```

7. Go the session by running the following command:

   ```
   cd('session')
   ```

   For example:

   ```
   cd('custom:/TopLink/TopLink:Name=Session(pricingui#V2.0file_/scratch/ri-
   user-1/data/pdc_domain/servers/AdminServer/tmp/_WL_user/pricingui_V2.0/
   jztz5c/APP-INF/lib/pricing_entities.jar_default)
   ```

8. Change the SQL and EclipseLink log level (as appropriate) by running the following command:

   ```
   set('CurrentEclipseLinkLogLevel',newLevel)
   ```

   Refer to Table 55-1 for a list of the different log levels and a brief description for each level.

Table 55-1 EclipseLink Log Levels

Level	Description
OFF	This setting disables the generation of the log output. You may want to set logging to OFF during production to avoid the overhead of logging.
SEVERE	This level enables reporting of failure cases only. Usually, if the failure occurs, the application stops.
WARNING	This level enables logging of issues that have a potential to cause problems. For example, a setting that is picked by the application and not by the user.

Table 55-1 (Cont.) EclipseLink Log Levels

Level	Description
INFO	This level enables the standard output. The contents of this output is very limited. It is the default logging level if a logging level is not set.
CONFIG	This level enables logging of such configuration details as your database login information and some metadata information. You may want to use the CONFIG log level at deployment time.
FINE	This level enables logging of the first level of the debugging information and SQL. You may want to use this log level during debugging and testing, but not at production.
FINER	This level enables logging of more debugging information than the FINE setting. For example, the transaction information is logged at this level. You may want to use this log level during debugging and testing, but not at production.
FINEST	This level enables logging of more debugging information than the FINER setting, such as a very detailed information about certain features (for example, sequencing). You may want to use this log level during debugging and testing, but not at production.
ALL	This level currently logs at the same level as FINEST.

Backing Up and Restoring PDC

This section describes the tasks that you can perform to back up and restore your PDC system.

To prevent any data loss and minimize the impact of software or hardware failure, back up your system immediately after installing or updating the system. Repeat the backup process whenever you make any changes in the data or in the configuration files.

If you do not back up the PDC system regularly, you need to reinstall and reconfigure PDC if the system is corrupted due to operational or system errors. Reinstalling and reconfiguring eliminates any chance of recovering and reprocessing data processed by the PDC system at the time of the error.

To perform a complete backup of your PDC system, make a complete offline copy of the following:

- The WebLogic server **domains** directory; by default, *MW_home***/user_projects/domains/PDC**. For instructions, see "Backing Up a Directory".
- The PDC installation directory and its content: *PDC_home*. *PDC_home* contains the PDC keystore. For instructions, see "Backing Up a Directory".
- The Oracle Inventory (**oraInventory**) directory. Open the **/etc/oraInst.loc** (Linux) file or the **/var/opt/oracle/oraInst.loc** (Solaris) file to find the default location of the **oraInventory** directory. For instructions, see "Backing Up a Directory".
- BRM Integration Pack and its content: *BRM_Integration_Pack_home*. This pack includes the keystore for BRM Integration Pack. For instructions, see "Backing Up a Directory".
- The PDC directory and its content: *PDC_BRM_home*. For instructions, see "Backing Up a Directory".

- The PDC database schema. Make a complete offline backup of your PDC database schema using the appropriate backup tools for your schema version.

  For example, on the machine on which the PDC users are created, run the Oracle database export (**exp**) utility to export the data in the PDC database to a file and then store it in a safe location. You can later run the import (**imp**) utility to import the data from the file into the PDC database when you restore PDC. See your database documentation for more information on performing full database backups.

- The transformation cross-reference database schema. Make a complete offline backup of your transformation cross-reference database schema using the appropriate backup tools for your schema version.

  For example, on the machine on which the PDC users are created, run the Oracle database export (**exp**) utility to export the data in the transformation cross-reference database to a file and store it in a safe location. You can later run the import (**imp**) utility to import the data from the file into the transformation cross-reference database when you restore PDC. See your database documentation for more information on performing full database backups.

- The migration cross-reference database schema, if you installed the migration utility during the PDC 11.1 Patch Set 4 or PDC 11.1 Patch Set 5 installation. Make a complete offline backup of your migration cross-reference database schema using the appropriate backup tools for your schema version.

  For example, on the machine on which the PDC users are created, run the Oracle database export (**exp**) utility to export the data in the migration cross-reference database to a file and store it in a safe location. You can later run the import (**imp**) utility to import the data from the file into the migration cross-reference database when you restore PDC. See your database documentation for more information on performing full database backups.

- The PDC database.

  Make a complete offline backup of your PDC database using the appropriate backup tools for your database version and ensure that the backup is completely valid and usable. The backup must contain both the database definition and all the database contents. See your database documentation for more information on performing full database backups.

> **Note:**
>
> Store this backup in a safe location. The data in these files will become necessary if you encounter any issues or during system failure.

Backing Up a Directory

To back up a directory:

1. Go to the directory for which you want to create a backup.
2. Copy the content of the directory to a new directory:

   ```
   cp -R DirectoryName NewName
   ```

   where *NewName* is the name for the new directory.

3. Create an archive of the entire directory:

   ```
   tar cvf NewName.tar NewName
   ```

   A compressed TAR file, of all copied files, is created with the extension **tar** (for example, weblogic_domain_bkp.tar).

4. Store the backup copy in a safe location (for example, in a different file system outside of PDC).

Restoring a Complete System Backup

To restore a complete PDC system backup:

1. On the machine in which PDC is installed, delete or rename the damaged directories.
   - To delete, run the following command:

     ```
     rm -r Directory_Name
     ```

   - To rename, run the following command:

     ```
     mv Directory_Name
     ```

2. Retrieve the backup TAR files.
3. Extract the backup copy of the directories from the TAR files:

   ```
   tar xvf Directory_Name.tar
   ```

   The command re-creates the directories in your restored installation directory.

4. Import the following schemas into an Oracle database or databases in the restored machine or in another machine:
   - PDC database schema
   - Transformation cross-reference database schema
   - Migration cross-reference database schema (if applicable)

   For example, if you have exported the data into a file by using the Oracle database export (**exp**) utility, run the Oracle database import (**imp**) utility to import the data from the file into an Oracle database. See your database documentation for more information.

5. Ensure that PDC is connected to the database or databases in which the schemas that you imported in step 4 are available by updating the database connection details in the following files:
   - *BRM_Integration_Pack_home***/apps/syncpdc/SyncPDCConfiguration.xml**
   - *PDC_home***/apps/bin/ImportExportConfiguration.xml**
   - *BRM_Integration_Pack_home***/apps/migration/MigrationConfiguration.xml**
   - *BRM_Integration_Pack_home*/**apps/transformation/TransformationConfiguration.xml**

6. Start all PDC processes.
7. Verify that the PDC configuration and the PDC data are recovered.

56
Troubleshooting Pricing Design Center

Learn how to troubleshoot problems with Oracle Communications Pricing Design Center (PDC) by using log files, and you can get help with PDC problems by contacting Oracle Global Support.

Topics in this document:

Troubleshooting Checklist

When any problems occur, it is best to do some troubleshooting before you contact Oracle Global Support:

- You know your installation better than Oracle Global Support does. You know if anything in the system has been changed, so you are more likely to know where to look first.
- Troubleshooting skills are important. Relying on Oracle Global Support to research and solve all of your problems prevents you from being in full control of your system.

If you have a problem with your PDC system, ask yourself these questions first, because Oracle Global Support will ask them of you:

- What exactly is the problem? Can you isolate it?

  Oracle Global Support needs a clear and concise description of the problem, including when it began to occur.
- What do the log files say?

  This is the first thing that Oracle Global Support asks for. Check the error log for the PDC component you are having problems with.
- Has anything changed in the system? Did you install any new hardware or new software? Did the network change in any way? Does the problem resemble another one you had previously? Has your system usage recently jumped significantly?
- Is the system otherwise operating normally? Has response time or the level of system resources changed? Are users complaining about additional or different problems?

Using Error Logs to Troubleshoot PDC

PDC error log files provide detailed information about system problems. If you are having a problem with PDC, look in the log files. PDC logs specific details about the PDC GUI and the transformation in separate sets of files.

PDC GUI Logs

PDC logs specific details about actions performed in the PDC GUI in Oracle WebLogic Server log files. See the Oracle WebLogic Server Administration Console Help for more information.

Transformation Engine Logs

PDC generates two types of transformation engine log files for each transformation engine:

- Master log file: This file logs the basic details of the transformation engine, such as the initialization details of the transformation engine, and so forth.
- Transaction log file: This file logs the details of the transactions performed by the transformation engine, such as work item details, details of calls to Oracle Communications Billing and Revenue Management (BRM) utilities, and so forth. A separate transaction log file is generated for every transaction.

The location of the transformation log files is specified in the *BRM_Integration_Pack_home***/apps/transformation/TransformationConfiguration.xml** file in the **<transformationLog>** elements, where *BRM_Integration_Pack_home* is the directory in which you installed the BRM Integration Pack.

You can define the logging level of the transformation engine log files in the *BRM_Integration_Pack_home***/apps/transformation/log-config.properties** file. The default logging level set in the file is **INFO**.

Diagnosing PDC Problems

PDC problems can be diagnosed using the Oracle WebLogic Server Diagnostic Framework. The Oracle WebLogic Server Diagnostic Framework allows you to collect, archive, and access diagnostic information about applications hosted on the WebLogic server. See the Oracle WebLogic Server Administration Console Help for more information.

Table 56-1 lists some common problems with PDC and shows you how to diagnose the error messages and resolve the problems.

Table 56-1 Common PDC Problems and Solutions

Problem	Possible Causes	Solution
Importing pricing components by using the **ImportExportPricing** utility fails.	Validation error due to one of the following reasons: • Incorrect configuration of the pricing components in the input XML file. • The input XML file does not conform to the format detailed in the XSD file for pricing components. • A feature configured for the pricing component is not supported by the target engine.	Check the **ImportExportPricing** utility log file for details of the error, correct the error, and import the pricing components again.
Changeset is successfully submitted through the PDC UI, but returns error.	There is invalid data that was not detected during validation.	Check the transformation engine log files for details of the error: • If the error message states that this is a "Configuration Missing" error, it is likely that a prerequisite setup component was not transformed successfully. Verify that all the referenced setup components are configured correctly. If required, update and resubmit the setup components and then use the **Fix** option to resubmit the changeset. • If there is nothing wrong with your configuration, verify that the target engine mappings (supported configurations) for the service, event, account, and RUMs used in the pricing components are correct. Ensure that the changes made in BRM to these components have been synchronized with PDC.
-	Error in the load utility of the target engine in BRM.	The error message will not provide the details of the error. Check the BRM **loadpricelist** and **LoadIFWConfig** utility log files for details of the error.
-	Transformation error	Transformation errors for pricing components submitted are returned in the PDC UI. Check the error messages in the PDC UI and follow the instructions below the error.
Error while submitting changeset with message "Error publishing changeset"	The job dispatcher is not configured correctly.	Check that the job dispatcher is deployed correctly and the status is Active in the WebLogic Administration Console. Check the Oracle WebLogic Server log files for relevant messages.
Changeset when submitted remains in pending state for a duration longer than normal.	Transformation engines are not running.	Check that the transformation engines are running on the system where BRM Integration Pack is installed by using the standard operating system commands.

Table 56-1 (Cont.) Common PDC Problems and Solutions

Problem	Possible Causes	Solution
-	Transformation engines are running but are configured to connect to the wrong WebLogic server instance.	Verify the host name of the WebLogic server in the *BRM_Integration_Pack_home***/apps/transformation/TransformationConfiguration.xml** file. If required, update the host name of the WebLogic server and restart the transformation engines.

SyncPDC is failing with error	Event attributes are not configured properly.	If the event attribute is defined in BRM, you must specify the following in the event definition file:
		`<targetApplicationSpecName>Pricing</targetApplicationSpecName>`
		`<targetApplicationSpecName>Billing</targetApplicationSpecName>`
		For more information, see "Enriching Event Definitions" in *PDC Creating Product Offerings*.

Getting Help for PDC Problems

If you cannot resolve the PDC problem, contact Oracle Global Support.

Before you contact Oracle Global Support, try to resolve the problem with the information logged in the log files. See "Using Error Logs to Troubleshoot PDC" for more information. If this does not help to resolve the problem, note the following information:

- A clear and concise description of the problem, including when it began to occur.
- Relevant portions of the relevant log files.
- Relevant configuration files.
- Recent changes in your system after which the problem occurred, even if you do not think they are relevant.
- List of all PDC components and patches installed on your system.

When you are ready, report the problem to Oracle Global Support.

57

Monitoring PDC with Prometheus and Grafana

Learn how to use external applications, such as Prometheus and Grafana, to monitor the system performance of Oracle Communications Pricing Design Center (PDC).

Topics in this document:

About Monitoring PDC

You set up the monitoring of PDC by using the following external applications:

- **WebLogic Monitoring Exporter**: Use this Oracle web application to scrape runtime information from PDC and then expose the metric data in Prometheus format. It exposes different WebLogic Mbeans metrics, such as memory usage and sessions count, that are required for monitoring and maintaining the PDC application.
- **Prometheus**: Use this open-source toolkit to scrape PDC metric data from WebLogic Monitoring Exporter and then store it in a time-series database.
- **Grafana**: Use this open-source tool to view on a graphical dashboard all PDC metric data that is stored in Prometheus.

Setting Up Monitoring in PDC

To set up monitoring in PDC:

1. Install the following external software on your system:
   - Prometheus. See "Installation" in the Prometheus documentation.
   - Grafana. See "Install Grafana" in the Grafana documentation.

   For the list of compatible software versions, see "Additional PDC Software Requirements" in *BRM Compatibility Matrix*.
2. Configure WebLogic Monitoring Exporter to scrape metric data from PDC. See "Configuring WebLogic Monitoring Exporter for PDC".
3. Configure Prometheus to scrape PDC metric data from WebLogic Monitoring Exporter. See "Configuring Prometheus for PDC".
4. Configure Grafana to display your PDC metric data in a graphic format. See "Creating Grafana Dashboards for PDC".

Configuring WebLogic Monitoring Exporter for PDC

To configure WebLogic Monitoring Exporter to scrape metric data from PDC:

1. Download the latest supported version of WebLogic Monitoring Exporter (**get*N.N*.sh**) from https://github.com/oracle/weblogic-monitoring-exporter/releases, where *N.N* is the version number.

   For the list of supported versions, see "Additional PDC Software Requirements" in *BRM Compatibility Matrix*.

2. Edit the **wls-exporter-config.yaml** file to include the PDC metrics that you want scraped. For the list of supported metrics, see "WebLogic-Based Application Metrics".

3. Run the following command:

   ```
   bash getN.N.sh wls-exporter-config.yaml
   ```

   The **wls-exporter.war** is downloaded to your current directory.

4. Deploy the **wls-exporter.war** to the server in which PDC is deployed.

WebLogic Monitoring Exporter is deployed on your servers and begins scraping metric data for PDC.

WebLogic Monitoring Exporter exposes your PDC metrics in Prometheus format to the **http://localhost:8080/wls-exporter/metrics** endpoint on the WebLogic Server.

Example 57-1 wls-exporter-config.yaml Entries for Scraping PDC

The following shows sample **wls-exporter-config.yaml** entries for setting up WebLogic Monitoring Exporter to scrape metrics from PDC:

```
metricsNameSnakeCase: true
queries:
- key: name
  keyName: location
  prefix: wls_server_
  applicationRuntimes:
    key: name
    keyName: app
    componentRuntimes:
      prefix: wls_webapp_config_
      type: WebAppComponentRuntime
      key: name
      values: [deploymentState, contextRoot, sourceInfo,
sessionsOpenedTotalCount, openSessionsCurrentCount,
openSessionsHighCount]
      servlets:
        prefix: wls_servlet_
        key: servletName
- JVMRuntime:
    prefix: wls_jvm_
    key: name
- executeQueueRuntimes:
    prefix: wls_socketmuxer_
    key: name
    values: [pendingRequestCurrentCount]
- workManagerRuntimes:
    prefix: wls_workmanager_
    key: name
```

```
    values: [stuckThreadCount, pendingRequests, completedRequests]
- threadPoolRuntime:
    prefix: wls_threadpool_
    key: name
    values: [executeThreadTotalCount, queueLength, stuckThreadCount,
hoggingThreadCount]
- JMSRuntime:
    key: name
    keyName: jmsruntime
    prefix: wls_jmsruntime_
    JMSServers:
      prefix: wls_jms_
      key: name
      keyName: jmsserver
      destinations:
        prefix: wls_jms_dest_
        key: name
        keyName: destination

- persistentStoreRuntimes:
    prefix: wls_persistentstore_
    key: name
- JDBCServiceRuntime:
    JDBCDataSourceRuntimeMBeans:
      prefix: wls_datasource_
      key: name
- JTARuntime:
    prefix: wls_jta_
    key: name
```

Configuring Prometheus for PDC

To configure a standalone version of Prometheus for PDC:

1. Configure Prometheus to scrape the PDC metrics exposed by WebLogic Monitoring Exporter.

   To do so, add a scraping configuration to your **prometheus.yaml** file that is similar to the one shown below for each managed server and admin server in your cluster.

   The following shows a sample admin server configuration for a PDC application that is deployed in the **pdc-domain** WebLogic domain:

   ```
   scrape_configs:
     - job_name: 'wls-metric-pdc'
       scrape_interval: 1s
       metrics_path: /wls-exporter/metrics
       scheme: http
       static_configs:
       - targets: ['hostname:port'] #hostname:port of the server where
   WebLogic Monitoring Exporter resides
         labels:
           weblogic_domainName: pdc-domain #domain name
           weblogic_domainUID: pdc-domain  #domain UID
           weblogic_serverName: AdminServer #server name
   ```

```
    basic_auth:
      username: username
      password: password
```

where *username* and *password* is your WebLogic Server user name and password.

2. Configure the alert rules in Prometheus.

   To do so, add alert rules to your **prometheus-rules.yaml** file that are similar to the ones shown below:

```
groups:
  -
    name: alertmanager.rules
    rules:
      - alert: heapFreeSizeWarning
        annotations:
          message: WLS heap free size is less than 60%.
        expr: wls_jvm_heap_free_percent{name="AdminServer"} <
60
        for: 2m
        labels:
          domain_uid: pdc-domain
          severity: critical
          wls_domain_namespace: {{namespace}}
      - alert: dataSourceReconnectWarning
        annotations:
          message: WLS PricingDB datasource reconnect failed.
        expr:
wls_datasource_failures_to_reconnect_count{name="PricingDB"} >
0
        for: 2m
        labels:
          domain_uid: pdc-domain
          severity: critical
          wls_domain_namespace: {{namespace}}
      - alert: WLSAdminServerNotRunningWarning
        annotations:
          message: WLS AdminServer stopped running.
        expr: wls_server_state_val{location="AdminServer"} !=
2
        for: 2m
        labels:
          domain_uid: pdc-domain
          severity: critical
          wls_domain_namespace: {{namespace}}
      - alert: PDCNotRunningWarning
        annotations:
          message: PDC stopped running.
        expr:
wls_webapp_config_deployment_state{location="AdminServer",app="Prici
ngDesignCenter",name="AdminServer_/pdc"} != 2
        for: 2m
        labels:
```

```
          domain_uid: pdc-domain
          severity: critical
          wls_domain_namespace: {{namespace}}
      - alert: PricingDBRunningWarning
        annotations:
          message: PricingDB stopped running.
        expr: wls_datasource_deployment_state{name="PricingDB"} !=
2
        for: 2m
        labels:
          domain_uid: pdc-domain
          severity: critical
          wls_domain_namespace: {{namespace}}
```

3. Configure Prometheus Alertmanager to raise alerts.

   To do so, add alert configurations that are similar to the ones shown below to your **prometheus.yaml** file.

```
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
       - localhost:9093
# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
  - "prometheus-rules.yaml"
```

4. Stop and restart Prometheus.

Creating Grafana Dashboards for PDC

You can create a dashboard in Grafana for displaying the metric data for PDC.

Alternatively, you can use the sample dashboards that are included with the PDC package. To use the sample dashboards, import the JSON file from the *PDC_home***/apps/Samples/monitoring** directory into Grafana. For information about importing dashboards into Grafana, see "Export and Import" in the *Grafana Dashboards* documentation.

> **Note:**
>
> For the sample dashboard to work properly, the datasource name for the WebLogic Domain must be **Prometheus**.

Part XI

ECE System Administration

This part describes Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) system administration tasks. It contains the following chapters:

58

ECE System Administration Overview

Learn about the system administration tasks you perform in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) and the tools you use to perform them.

> **Caution:**
>
> Deploying charging for 5G with HTTP Gateway (5G CHF) requires a cloud native deployment of ECE and BRM components. The HTTP Gateway can be used only on an ECE cloud native system.

Topics in this document:

ECE System Administration Tasks and Applications

Administrating ECE includes the following tasks:

- Starting and stopping ECE nodes. See "Starting and Stopping ECE ".
- Monitoring the system and managing the components. See "Monitoring ECE Using ECE Monitoring Agent".
- Configuring installed components. See "Configuring JVM Tuning Parameters".
- Troubleshooting. See "Troubleshooting ECE ".

To administer ECE, use the following applications:

- Elastic Charging Controller to start and stop nodes, add and remove nodes, and perform troubleshooting tasks such as simulating usage charging. ECC is an enhancement of the Groovy Shell command line. See "Running Elastic Charging Controller (ECC)".
- Java Monitoring and Management Console (JConsole) or any JMX client to change ECE configuration parameters by editing MBean attributes. See the individual ECE configuration procedures.

Running Elastic Charging Controller (ECC)

Use ECC to do the following:

- Add and remove nodes in the topology
- Start and stop data nodes
- Check the status of a node

- Perform system management tasks (for example, use the **infoCollector** utility to collect diagnostic information)

To run ECC:

1. On the driver machine, path to the **/oceceserver/bin** directory.
2. Start Elastic Charging Controller (ECC).

   ```
   ./ecc
   ```

3. Run the ECC commands; for example:

   ```
   start server
   ```

ECC uses standard Groovy Shell commands, such as **help** and **exit**. You can complete commands by pressing **space bar** and then **Tab key**.

See "ECC Commands" for more information.

59

Setting Up and Managing Elastic Charging Engine Security

Learn how to implement security in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) by setting up user accounts, user groups, and passwords.

Topics in this document:

See also "Enabling Secure Communication between BRM Components" and *BRM Security Guide*.

Setting Up User Accounts and User Groups

Access to ECE files is controlled by creating user accounts and granting specific permissions in the Elastic Charging Controller (ECC). After you have created user groups and set permissions, users can log in through ECC and manage ECE files.

Create the following user accounts to manage ECE files:

- UNIX accounts:
  - An administrator to run and manage ECE processes
  - If needed, users to manage the rated event files
  - If needed, users to manage ECE file systems
- Non-UNIX accounts:
  - Users created for securing ECE processes from unauthorized access.

You must also create a UNIX user group that includes all users, including the administrator.

For information about the permissions that you should grant these users and groups, see "Managing ECE Permissions".

Creating UNIX User Accounts and Groups

Create UNIX accounts and groups using UNIX commands in ECC. To create user accounts, use the **useradd** command. To create groups, use the **groupadd** command.

Managing ECE Permissions

This section describes the permissions that you should restrict for a secure ECE installation. Strict governance of file permissions prevents accidental overwriting and misuse of the system.

About Permission Types

Access to ECE files is controlled by creating user accounts and granting specific permissions in ECC. For information about which user accounts to create, see "Setting Up User Accounts and User Groups". After you have created user groups and set permissions, users can log in through ECC and manage ECE files.

ECE uses the traditional UNIX read, write, and execute permission types. In ECC, use the **chmod** command to set file permissions and the **chown** command to change file owners.

The **chmod** command can change the permissions of files and directories that have been created. You use the **setfacl** command to set permissions for files that have not been created at the time of installation, such as log files in the *ECE_home***/logs** directory.

Restrict permissions as much as possible. You may choose to have either a single administrative user with all permissions who runs ECE core processes and manages the rated event files and other directories, or to create multiple users with specific permissions to carry out these tasks.

Configuring Specific File Permissions

To ensure the integrity of the system, access to certain files should be tightly regulated.

Restrict access to the following files:

- **jmxremote.password**: Ensure this file has read permission for the user and no permission to either the group or the world (Unix 400).
- **server.jks**: Ensure only you have read permission.
- **charging-cache-config.xml**: Ensure this file has only read permission for the user.
- **charging-coherence-override-*mode*.xml**: Ensure this file has only read permission for the user.

To restrict access to these files:

1. Log in to ECC.
2. Go to the *ECE_home***/config** directory.
3. Enter the following commands:

```
chmod 400 jmxremote.password
chmod 400 server.jks
chmod 400 charging-cache-config.xml
chmod 400 charging-coherence-override-mode.xml
```

Granting ECE User Permissions

Follow these guidelines when granting user permissions:

- Grant the administrator user all permissions.
- Grant execute permissions to only the user assigned to start processes.

> **Note:**
>
> On Linux, all directories must have an execute permission to the user.

- If you created users to manage the rated event files, grant them read and write permissions for the rated event directory.
- If you created users to manage file systems, grant them read and write permissions to the directories that they will manage.

To grant permissions to users, use the **chmod** command in ECC. To change file owners, use the **chown** command in ECC. To change the permissions for files, use the **setfacl** command.

The UNIX group containing all of the users should have the following directory permissions:

- Read and write permission to *ECE_home***/config** and *ECE_home***/logs**
- Read permission to *ECE_home***/lib**
- Read and execute permission to *ECE_home***/bin**

About the Default umask for ECE Users

You control permissions for new files by adjusting the default **umask**. You check your current **umask** by entering the **umask** command with no additional parameters. Users can also change their **umask** if they wish.

The default **umask** is set to **007** to exclude all users that are not in the same group.

If no default **umask** is set, any new files will be created with read-write-execute permissions granted to all users. Instead of having a **umask** that changes the default permissions globally, run the **setfacl** command to set permissions for files. For example, to set all logs in the logs directory to **600** for the logs that are created and logs that will be created in the future:

```
setfacl -m user::rw-,group::---,other::--- ECE_HOME/logs/*.log
```

About External Permissions

Permissions for non-UNIX accounts are defined within the **permissions.xml** file. When you install ECE, you create a principal administrator alias and password for Oracle Coherence cluster security, which is granted all permissions. The following example shows the content of **permissions.xml** after installation:

```
<permissions>
  <grant>
    <principal>
      <class>javax.security.auth.x500.X500Principal</class>
      <name>CN=Administrator,OU=DN</name>
    </principal>
```

```
    <permission>
      <target>*</target>
      <action>all</action>
    </permission>
  </grant>
</permissions>
```

where *Administrator* and *DN* are the DName credentials that were entered in the KeyStore Credentials screen at installation.

Managing Passwords in ECE

Access to ECE files is controlled by creating user accounts and granting specific permissions in ECC. For information about which user accounts to create, see "Setting Up User Accounts and User Groups". For information about granting permissions, see "Managing ECE Permissions".

Each UNIX-based user account is password protected and can be managed through ECC. These passwords do not expire by default. You can choose to set up password expiration using the **chage** command.

In addition to UNIX-based user accounts, you also create a non-UNIX account for access to external applications. ECE connects to Oracle Communications Billing and Revenue Management (BRM) and Pricing Design Center (PDC) to load and store the pricing and customer data required to charge for network events. For secure communication between these applications, credentials are stored in the Oracle wallet. See "About Managing External Application Passwords".

The passwords from ECE used to integrate with external systems do not expire by default. Change the secret keys used to encrypt them regularly.

About Managing External Application Passwords

When installing ECE, the passwords that you enter for Oracle Coherence security are automatically encrypted. The encryption keys are stored in the **server.jks** file, and the password to the keys are stored in the **jmxremote.password** file. The **server.jks** file is created in *ECE_home*/**config** when you install ECE. You can check the validity of the KeyStore. See "Checking Keystore Validity" for more information.

The passwords that you enter for the boundary systems (such as BRM and PDC) are stored in the Oracle wallet when you install ECE. If the password used to connect to BRM or PDC is changed or if a new BRM or PDC instance is added, you can add or modify the password in the ECE wallet by using ECE Mbeans. See "Storing or Modifying Passwords in ECE Wallet" for more information.

Storing or Modifying Passwords in ECE Wallet

To store or modify a password in the ECE Oracle wallet:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand the node of the component and instance for which you want to set or modify the password.

For example, if you want to modify the BRM password, expand **charging.connectionConfigurations.brmConnection**.

4. Expand **Operations**.
5. Click **SetPassword**.
6. Specify the following values:
   - **password.** Enter the password you want to add or modify.
   - **walletPassword.** Enter the password of the ECE Oracle wallet.
7. Click the **SetPassword** button.

Checking Keystore Validity

To check keystore validity:

1. Log in to ECC.
2. Enter the following commands:

```
keytool -list -v -keystore server.jks
password: storepassword
```

   where *storepassword* is the password to the keystore.

   ECC returns the contents of **server.jks**, indicating that the keystore is valid.

If ECC denies permission, ensure that you have set the permissions to **server.jks** as described in "Configuring Specific File Permissions".

If ECC does not return the contents of the JKS file, it could be because the user forgot the password. Access will be denied until the ECE user goes through the company approved process for recreating the keystore with appropriate accounts.

The ECE installer creates a self signed certificate. Your company's authorized personnel can create a certificate and get it signed by a signing authority or self sign it. Deployment of all keystores must be validated for authenticity.

Setting Up Cluster Security

This section explains how to perform tasks related to ECE cluster security. To restrict access to the ECE cluster (an Oracle Coherence cluster), you must set up an authorized hosts list. You can optionally enable SSL for intra-cluster communication, in which case you must also enable Well Known Addresses (WKA).

Adding Trusted Hosts

The trusted host list is set up at installation in the Coherence Grid Security screen of the Installer. This list allows only specified hosts to connect to the cluster. You can add more hosts to the list in the charging override file.

The override file contains a section **<authorized-hosts-list>** that contains host address of each physical machine participating in the cluster. The address could be a server host name or an IP address. IP address is preferred if the hosts are multi-homed. Only the IP address that the network is binding to should be given. Using IP addresses also allows you to set a range of IP addresses which is not possible when using host names.

To add trusted hosts:

1. Open the *ECE_home*/**config/charging-coherence-override-secure-prod.xml** file in a text editor.
2. In the trusted hosts list, add new trusted hosts below those specified at installation by adding the following text:

```
<cluster-config>
  <authorized-hosts>
    <host-address>host_ip1</host-address>
    <host-address>new_host_ip</host-address>
    <host-range>
      <from-address>ip_range_start1</from-address>
      <to-address>ip_range_end1</to-address>
    </host-range>
    <host-range>
      <from-address>new_ip_range_start</from-address>
      <to-address>new_ip_range_end</to-address>
    </host-range>
  </authorized-hosts>
</cluster-config>
```

   where:

   - *host_ip1* is the pre-existing IP address of a host specified at installation.
   - *new_host_ip* is a specific host that you are adding.
   - *ip_range_start1* is the beginning of a pre-existing range of IP addresses specified at installation.
   - *ip_range_end1* is the end of the pre-existing range of IP addresses specified at installation.
   - *new_ip_range_start* is the beginning of the new range of IP addresses.
   - *new_ip_range_end* is the end of the new range of IP addresses.

   You can choose to enter *new_host_ip*, both *new_ip_range_start* and *new_ip_range_end*, or all three.
3. Save and close the file.

Securing Intra-Cluster Communication

Intra-cluster communication can be strengthened by enabling SSL. ECE supports SSL versions 2 and 3 and TLS version 1. For an overview of common SSL concepts, see *Oracle Coherence Security Guide*.

You should expect enabling SSL to have a negative impact on ECE performance. As such, SSL should only be enabled if required. Consider the implications of the performance degradation before enabling intra-cluster SSL.

By default, SSL is not enabled. You can select to enable it when you run the ECE installer. You can also enable SSL after installing ECE. See "Performing a Secure ECE Installation" in *BRM Security Guide*.

Enabling SSL Within the ECE Cluster

To enable SSL:

1. Enable WKA. See "Enabling Well Known Addresses".
2. Open the *ECE_home*/**config/ece.properties** file.
3. Search for the following system property:

   ```
   tangosol.coherence.override=
   ```

4. Add **charging-coherence-override-secure-prod.xml** as follows:

   ```
   tangosol.coherence.override=charging-coherence-override-secure-prod.xml
   ```

5. Open the *ECE_home*/**config/defaultTuningProfile.properties** file.
6. Search for the following system properties:

   ```
   tangosol.coherence.ssl.storepassword=
   tangosol.coherence.ssl.keypassword=
   ```

7. Set the properties as follows so that two-way SSL can use **server.jks** for encrypted data:

   ```
   tangosol.coherence.ssl.storepassword=keystorepassword
   tangosol.coherence.ssl.keypassword=coherencepassword
   ```

   where:

   - *keystorepassword* is the password to **server.jks**
   - *coherencepassword* is the password for Oracle Coherence grid security alias that was entered in the KeyStore Credentials screen at installation

8. Save and close the files.

To generate an SSL self-signed certificate:

1. Log in to the ECC.
2. Run the following command:

   ```
   keytool -genkeypair -dname "cn=Administrator, ou=DN" -alias admin -keypass
   password -keystore ECE_home/config -storepass storepassword
   ```

   where:

   - *Administrator* and *DN* are the DName credentials that were entered in the KeyStore Credentials screen at installation
   - *password* is the key password for the admin alias
   - *storepassword* is the password for the KeyStore

   You can choose to set certificate expiry using the **-validity** option.

You can optionally have the certificate signed by a signing authority, but for ECE functionality a self-signed certificate is sufficient.

Enabling Well Known Addresses

The Well Known Addresses (WKA) mechanism allows cluster members to discover and join a cluster using unicast instead of multicast. You may need to enable WKA if your data center policy prohibits multicast. If you enable SSL for intra-cluster communication, you must enable WKA.

To enable WKA:

1. Open the *ECE_home*/**config/charging-coherence-override-secure-prod.xml** file.

2. Specify cluster members by adding text as follows:

```
<cluster-config>
  <unicast-listener>
    <well-known-addresses>
      <socket-address id="id">
        <address>ip_address</address>
        <port>port</port>
      </socket-address>
      ...
    </well-known-addresses>
  </unicast-listener>
</cluster-config>
```

where:

- *id* is the ID for a particular cluster member
- *ip_address* is the IP address of the cluster member
- *port* is the value specified in the member's unicast listener port

Ensure that the list of WKA members is the same on every cluster member so that no cluster member operates independently from the rest of the cluster.

For more information about setting up a WKA host list, see the discussion of Well Known Addresses in *Oracle Coherence Developer's Guide*.

Securing Inter-Cluster Communication

Inter-cluster communication between the ECE cluster and the BRM system can be strengthened by enabling SSL.

BRM Gateway can connect to the Connection Manager (CM) by using SSL. See "Enabling SSL Communication between BRM Gateway and the CM" for information.

Enabling SSL Communication between BRM Gateway and the CM

If you use SSL to secure connections between Java PCM clients and the CM, you must enable SSL communication between BRM Gateway and the CM.

For information about enabling SSL for Java PCM clients in BRM, see "Enabling SSL/TLS for Java PCM Clients".

To enable SSL communication between BRM Gateway and the CM:

1. Do one of the following, where *BRM_home* is the directory in which you installed BRM:
   - If you enabled one-way SSL server and client authentication in the CM, verify that the *BRM_home***/wallet/client/cwallet.sso** file was copied into your ECE installation (in the directory of your choice).
   - If you enabled two-way SSL server and client authentication between the CM and its PCM clients, verify that the *BRM_home***/wallet/server/cwallet.sso** file was copied into your ECE installation (in the directory of your choice).

     For information about enabling two-way SSL server and client authentication between the CM and its PCM clients, see "Enabling SSL/TLS in Connection Managers".

2. If the file listed in the previous step is not in your ECE installation, copy it from BRM to the ECE directory of your choice.
3. Copy the following JAR files to the *ECE_home*/**lib** directory:
   - *BRM_home*/**jars/osdt_cert.jar**
   - *BRM_home*/**jars/osdt_core.jar**
   - *BRM_home*/**jars/oraclepki.jar**
4. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
5. Expand the **ECE Configuration** node.
6. Expand **charging.connectionConfigurations.brmConnection**.
7. Expand **Attributes**.
8. Specify values for the following attributes:
   - **sslEnabled**: Change the value from **0** to **1**.
   - **wallet**: Enter the path to the directory on your ECE installation where you copied your Oracle wallet from your BRM environment (**cwallet.sso**).
9. Stop and restart BRM Gateway.

   See "Starting and Stopping the BRM System" for information.

SSL communication between BRM Gateway and the CM is now enabled.

Setting Up Password-less SSH Between the Driver and Servers

You must set up bi-directional password-less Secure Shell (SSH) logins between the driver machine and each server machine for ECC to work. Password-less SSH allows servers to connect to the driver and synchronize ECE files.

To set up password-less SSH:

1. Log in to the ECC.
2. Run the following commands:

   ```
   ssh-keygen -t dsa
   ssh-copy-id -i ~/.ssh/id_dsa.pub user@host
   ```

   where:

   - *user* is the user name set for all host machines at installation in the ECE Cluster Details screen.
   - *host* is the name of the server for which the password-less SSH is being established.

To test passwordless SSH:

1. Log in to ECC.
2. Run the following command:

   ```
   ssh user@host
   ```

   where:

   - *user* is the user name set for all host machines at installation in the ECE Cluster Details screen.

- *host* is the name of the server for which the password-less SSH is being established

If a password is requested, password-less SSH setup has failed. Ensure that you have followed the steps for setting up password-less SSH correctly and run the test again.

60

Starting and Stopping ECE

Learn how to start, stop, and restore Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

> **Caution:**
>
> Stopping the ECE system is not expected. Stopping all charging server nodes results in data loss. Do not stop all charging server nodes unless you want all data to be removed from the ECE caches. If you stop all charging server nodes, you cannot restore the cached data.

About Starting and Stopping ECE

You start and stop ECE by using Elastic Charging Controller (ECC) **start** and **stop** commands on ECE nodes. The syntax is:

```
start [role | node_name]

stop [role | node_name]
```

where:

- *role* is the node that you want to start.
- *node_name* is the name of the node as you defined it in the node-name column of the *ECE_home*/**config/eceTopology.conf** file.

For example:

- This command starts all nodes that have the **server** role; that is, all charging server nodes:

  ```
  start server
  ```

- This command starts all Diameter Gateway instances:

  ```
  start diameterGateway
  ```

- This command starts the specific Diameter Gateway instance:

  ```
  start diameterGateway1
  ```

- This command starts a node running Pricing Updater:

  ```
  start pricingUpdater
  ```

- This command starts the charging server node named **ecs1**:

  ```
  start ecs1
  ```

- The command with no role or node name specified starts or stops all charging server nodes:

  ```
  start
  ```

Roles and node names can be seen by using a JMX editor to display MBeans.

Starting ECE

Before starting ECE, verify that the NoSQL database is running. The NoSQL database must be running before you start Rated Event Formatter. See the NoSQL database documentation.

To start all of the ECE nodes:

1. On the driver machine, go to the **/bin** directory.
2. Start Elastic Charging Controller (ECC).

   ```
   ./ecc
   ```

3. Run the following commands in this order:

   ```
   start
   start configLoader
   start pricingUpdater
   start customerUpdater
   ```

4. Verify that the ECE nodes are in a usage processing state:
   a. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
   b. Expand the **ECE State Machine** node.
   c. Expand **StateManager**.
   d. Expand **Attributes**.
   e. Make sure the **stateName** attribute is set to **UsageProcessing**.
5. Run this command to start EM Gateway:

   ```
   start emGateway
   ```

6. Run this command to start BRM Gateway:

   ```
   start brmGateway
   ```

7. Run this command to start Rated Event Formatter:

   ```
   start ratedEventFormatter
   ```

8. Run this command to start Diameter Gateway:

   ```
   start diameterGateway
   ```

9. Run this command to start RADIUS Gateway:

   ```
   start radiusGateway
   ```

Stopping ECE

> **Note:**
>
> Stopping charging server nodes removes all data from memory. For example, if you run **stop server**, you will lose all cached data.

Non-charging server nodes can be stopped and restarted with no loss of cache data. For example, stopping the following ECE nodes temporarily disrupts real-time data updates coming from other applications, but the updates are processed upon restart of the ECE system:

- Customer Updater

  ```
  stop customerUpdater
  ```

- Pricing Updater

  ```
  stop PricingUpdater
  ```

- BRM Gateway

  ```
  stop brmGateway
  ```

- External Manager (EM) Gateway

  ```
  stop emGateway
  ```

Restoring the ECE System

Restarts of the ECE system are not expected because ECE is built to always be running. ECE has built-in high availability and fault tolerance when server redundancy is employed. Server redundancy is a minimum requirement of ECE installations.

If you stop all charging server nodes on a standalone system, you will lose data in Coherence caches.

Patching ECE to upgrade it to a new version does not require restarting the system when you perform a rolling upgrade. See "Upgrading Existing ECE 12.0 Installation" in *ECE Installation Guide* for more information about rolling upgrades.

If a restart is required after all ECE nodes are stopped, restore the ECE system by doing the following:

1. In PDC, publish all the PDC pricing data (the metadata, setup, pricing, and profile data) from the PDC database to ECE by running the following command:

   ```
   ImportExportPricing -publish -metadata -config -pricing -profile -target [ece]
   ```

2. In ECE do the following:
   a. On the driver machine, change to the *ECE_home*/**bin** directory.
   b. Start ECC.

      ```
      ./ecc
      ```

   c. Start ECE processes and gateways in the following order:

> **Note:**
>
> Depending on your installation, you can start Diameter Gateway, RADIUS Gateway, both Diameter Gateway and RADIUS Gateway, or none.

```
start server
start configLoader
start pricingUpdater
start customerUpdater
start emGateway
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
```

All data is now back in the ECE data grid.

Real-time data updates (coming from BRM) that had been temporarily disrupted due to the shutdown are processed upon restart.

Diameter Gateway processing that had been temporarily disrupted due to the shutdown is resumed upon restart. For example, the translation of Diameter requests into ECE Java API requests and the processing of push notifications from Elastic Charging Server (the processing of ECE JMS notifications) is resumed.

Troubleshooting Starting ECE

If the **start** command fails when starting ECE in a large cluster, the problem may be due to the limitation on how many simultaneous SSH connections the machines in your environment can make from another machine or to another machine. The **start** command, by default, attempts to start ten nodes simultaneously using ten different threads. The **start** command might file in the following cases:

- If the maximum number of open sessions permitted per network connection is less than ten on the driver machine.
- If the maximum number of simultaneous connections to the SSH daemon on the server machine is less than the number of threads attempting to make connections on the server machine.

To resolve the issue, change the number of nodes the **start** command will start simultaneously, or (if your runtime environment allows), change the limit of simultaneous SSH connections your environment can make from or to a machine.

To set the limit for how many simultaneous SSH connections can be made from the driver machine in your environment:

1. On the driver machine, stop the SSH daemon.
2. Open the **/etc/ssh/sshd_config** file.
3. Set the **MaxSessions** property.

   MaxSessions specifies the maximum number of open sessions permitted per network connection. The default is **10**.

4. Save and exit the file.
5. Start the SSH daemon.

To set the limit for how many simultaneous SSH connections can be made to the server machine in your environment:

1. On the server machine, stop the SSH daemon.
2. Open the **/etc/ssh/sshd_config** file.
3. Set the **MaxStartups** property to be equal or more than the number of nodes the start command starts at one time.

   MaxStartups specifies the maximum number of concurrent unauthenticated connections to the SSH daemon on the server machine. Additional connections are dropped until authentication succeeds or the LoginGraceTime expires for a connection. The default is **10**.
4. Save and exit the file.
5. Start the SSH daemon.

To set the number of nodes the start command will start simultaneously:

1. On the driver machine, open the *ECE_home***/config/ece.properties** file.
2. Add the following line to the file:

   ```
   numberOfStartCommandsExecutorThreads=number_of_threads
   ```

   where *number_of_threads* is the number of threads the start command uses to start the (same) number of nodes simultaneously. It must be equal to or less than the value of the **MaxSessions** property set for the given environment.
3. Save and close the file.

61

Monitoring ECE Using ECE Monitoring Agent

Learn how to monitor and manage Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) using the ECE Monitoring Agent.

Topics in this document:

About Monitoring ECE Using Monitoring Agent

You can use the ECE Monitoring Agent to monitor:

- **ECE cluster.** Monitors the state of each ECE node configured with JMX port in the ECE topology and generates alerts in case of node failures, shutdowns, or network failures. For ECE charging server nodes, the Monitoring Agent generates alerts when there are unbalanced partitions across ECE server nodes and when the number of running storage-enabled nodes falls below the threshold configured.
- **ECE caches.** Monitors the ECE cache based on the estimated size of the cache and the threshold configured and generates alerts when the threshold breach occurs. You can define the threshold for generating alerts; for example, you can configure the threshold percentage to generate an alert when the threshold is approaching or when the threshold breach occurs.

  You can configure to monitor the following caches:

  - Subscriber-based caches, such as Customer and Balance caches, based on the estimated size and the number of nodes available.
  - Session-based caches, such as ActiveSession and RatedEvent caches, based on the number of nodes configured and the size of each node.
- **WebLogic Server and Oracle NoSQL database connections.** Monitors the connections to WebLogic Server and the Oracle NoSQL database storage nodes and reports connection failures encountered while publishing notifications or storing rated events into the database.
- **ECE server response time.** Monitors the ECE server response time for the different types of ECE requests received during the monitoring cycle; for example, IUT requests. You can define the response time and configure the Monitoring Agent to generate alerts in case of threshold breaches. The information about the requests are included in the

DiameterGatewayStatistics and **EMGatewayStatistics** sections in the summary report logs.

- **Rated event throughput.** Monitors the number of rated events stored or retrieved from the cache or from the Oracle NoSQL database and generates alerts on threshold breaches.
- **Charging sessions.** Monitors the status of all charging sessions and logs the total number of sessions that are opened or closed after the last monitoring check and the sessions that are currently active at the time of current monitoring check. This summary includes the overall status of the charging sessions.
- **Network sessions.** Monitors the status of all network sessions and logs the total number of sessions that are opened or closed after the last monitoring check and the sessions that are currently active at the time of the current monitoring check. This summary includes the overall status of the network sessions.
- **Diameter Peers.** Monitors all the Diameter peers connected to the Diameter Gateway instances at regular intervals and logs the details of the peers (including the transport protocol). You can also view the details of the diameter peers that are connected to specific Diameter Gateway instances by using ECE MBeans. See "Viewing Active Diameter Peers" in *ECE Implementing Charging*.
- **Diameter Responses.** Monitors all the Gy and Sy responses and their result codes and logs the total number of success and failure responses. This summary includes the overall status of the responses and the details of the failed responses for each result code and product type. The product type is a combination of Service-Context-Id, Service-Identifier, and Rating-Group and it is shown in the summary only for the Initiate, Update, Terminate (IUT) requests. For other requests, such as TopUp, Debit, and Refund, the product type is not applicable and it is shown as **Unknown**.

In addition to generating periodical reports and alerts on threshold breaches, the Monitoring Agent reports any unusual scenarios that prevents ECE from continuing the processes. By default, the Monitoring Agent generates the following types of alerts: Warning, Critical, and Fatal. You can also define the conditions for generating the alerts; for example, you can define to generate alerts only five times in an hour.

To configure monitoring of ECE, a sample file named **monitor-configuration.xml**, is included in the *ECE_home***/config** directory. You can use this file to configure the settings for monitoring ECE. See "Configuring Monitoring of ECE" for more information.

The Monitoring Agent runs periodically based on the configured intervals. It also runs when certain event-based scenarios occur; for example, node failures.

Types of Log Files

The Monitoring Agent generates the following types of log files:

- monitorAgentX.log
- monitorAgentX_ECE_SUMMARY_REPORT.log
- monitorAgentX_ECE_ALERT_REPORT.log

These log files are stored in the *ECE_home***/logs** directory by default. Review these log files regularly to monitor your system and detect and diagnose system problems.

monitorAgentX.log

This log file contains general information about various activities of the Monitoring Agent. This log provides information about the issues encountered by the Monitoring Agent.

monitorAgentX_ECE_SUMMARY_REPORT.log

This log file contains the summary report generated at regular monitoring intervals in the JSON format. The last section in the log file contains the details and the format of the information provided in the summary logs. By default, the Monitoring Agent creates 24 summary log files in a day (one summary log file for each hour of the day). After 24 hours, it starts deleting the summary log files one by one every hour and adds the summary log files for the current day. You can view a maximum of 24 summary log files at any time of a day.

monitorAgentX_ECE_ALERT_REPORT.log

This log file contains the alerts logged at regular monitoring intervals in the JSON format. The Alert Format section in this file specifies the formats and the types of alerts logged. The Monitoring Agent publishes the alerts as notifications to a JMS queue. By default, the Monitoring Agent creates 24 alert log files in a day (one alert log file for each hour of the day). After 24 hours, it starts deleting the alert log files one by one every hour and adds the alert log files for the current day. You can view a maximum of 24 alert log files at any time of a day.

You can also view these notifications and subscribe to receive the notifications by accessing the **ECE Monitoring.Notifier** node. See "Subscribing Notifications" for more information.

Configuring Monitoring of ECE

To configure monitoring of ECE:

1. Open the *ECE_home*/**config/monitor-configuration.xml** file.
2. Specify or update the value for the entries listed in Table 61-1 as appropriate.

Table 61-1 Entries in the Monitoring Configuration XML File

Command	Description
nodeHealthCheckInterval	The regular interval (in seconds) in which the Monitoring Agent is run. **Note:** The maximum and the default interval is 30 seconds.
alertCountResetInterval	The interval (in seconds) for resetting the alert count.
fatalAlertCount	The number of fatal alerts generated after which the alerts are suppressed for the time defined by **alertCountResetInterval**.
criticalAlertCount	The number of critical alerts generated after which the alerts are suppressed for the time defined by **alertCountResetInterval**.
warningAlertCount	The number of warnings generated after which they are suppressed for the time defined by **alertCountResetInterval**.

Table 61-1 (Cont.) Entries in the Monitoring Configuration XML File

Command	Description
iutThresholdLatency	The latency (in milliseconds) for the Initiate, Update, and Terminate requests from Diameter Gateway to the ECE server. **Note:** Ensure that you enter the appropriate latency for generating this report; for example, a 99.99% or 100% latency. When **iutThresholdLatency** is set, the Monitoring Agent tracks the percentage of requests that are below the **iutThresholdLatency** value and generate the alerts based on **alertType** and **alertValue**. For example, if **iutThresholdLatency** is set to 10 and out of 100 calls made, 95 calls are less than or equal to 10 milliseconds, the latency percentile is considered as 95. If **alertType** is set to Warning and **alertValue** for Warning is set to 95, the Monitoring Agent checks if the latency percentile is less than **alertValue**, which is 95. In this case, the latency percentile is not less than **alertValue** for Warning, therefore the Monitoring Agent does not generate a Warning alert. In case if the latency percentile is 94, the Monitoring Agent generates a Warning alert.
thresholdLatency	The latency (in milliseconds) for all the traced requests from EM Gateway to the ECE server. When **thresholdLatency** is set, the Monitoring Agent tracks the percentage of requests that are below the **thresholdLatency** value and generates alert based on the **alertType** and **alertValue**. For example, if **thresholdLatency** is set to 10 and out of 100 requests received, 89 requests took less than or equal to 10 milliseconds, the latency percentile is considered as 89. If **alertType** is set to Critical and **alertValue** for Critical is set to 90, the Monitoring Agent checks if the latency percentile is less than **alertValue**, which is 90. In this case, the latency percentile is less than **alertValue** for Critical, therefore the Monitoring Agent generates a Critical alert. If the latency percentile is equal to or more than **alertValue**, the Monitoring Agent does not generate a Critical alert. However, if the latency percentile is less than the **alertValue** for Warning, the Monitoring Agent generates a Warning alert.
alertType	The type of alert generated. The following are the valid alert types: Warning, Critical, and Fatal.

Table 61-1 (Cont.) Entries in the Monitoring Configuration XML File

Command	Description
alertValue	The value at which the alert is generated. You specify the value for this entry as follows: • In the **diameterGatewayLatency** and **emGatewayLatency** sections, you specify the percentages for tracking requests from Diameter Gateway and EM Gateway. For example, you can configure 95% latency for Warnings, 90% latency for Critical alerts, and 80% latency for Fatal alerts. • In the **ratedEventThroughput** section, you specify the ratio for tracking rated events throughput. This is the ratio of the number of rated events stored or retrieved from the cache or from the Oracle NoSQL database. For example, you can configure 0.95 for Warnings, 0.90 for Critical alerts, and 0.80 for Fatal alerts. • In the **partitionsUnbalanced** section, you specify the number of occurrences for tracking unbalanced partitions across ECE server nodes. For example, you can configure 3 occurrences for Warnings, 10 occurrences for Critical alerts, and 20 occurrences for Fatal alerts. • In the **runningServers** section, you specify the number of server nodes that are currently running for tracking the server nodes availability. For example, you can configure 2 nodes for Warnings, 1 node for Critical alerts, and 0 node for Fatal alerts. • In the **noSQLCommitFailure** and **webLogicPublishFailure** sections, you specify the number of connection failures at which the alerts are generated. For example, you can configure 1 failure for Warnings, 3 failures for Critical alerts, and 5 failures for Fatal alerts. • In the **subscriberCacheUtilization** section, you specify the percentage for tracking the total subscriber cache size (in bytes) across all the ECE server nodes; for example, reaching 60% of cache size for Warnings, reaching 80% of cache size for Critical alerts, and reaching 100% of cache size for Fatal alerts. • In the **sessionCacheUtilization** section, you specify the percentage for tracking the total session cache size (in bytes) for a ECE server node; for example, reaching 60% of cache size for Warnings, reaching 80% of cache size for Critical alerts, and reaching 100% of cache size for Fatal alerts.
subscriberCacheCapacity name	The name of the subscriber-related caches, such as Customer and Balance caches.
projectedClusterCapacity	The projected capacity of all ECE server nodes to hold a configured subscriber-based cache, such as Customer and Balance caches.
sessionCacheCapacity name	The name of the session-related caches, such as ActiveSession, RatedEvent, ServiceContext, and RecurringBundleIdHistory.
projectedNodeCapacity	The projected capacity of a ECE server node to hold a configured session-based cache, such as the ActiveSession cache.

3. Save and close the file.
4. On the machine where you have Elastic Charging Controller (ECC) installed, go to *ECE_home*/**bin**.
5. Start ECC:

   ```
   ./ecc
   ```

6. Run the following command, which deploys the ECE installation onto the server machines:

   ```
   sync
   ```

The **sync** command copies the relevant files of the ECE installation onto the server machines in the ECE cluster.

7. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
8. Expand the **ECE Monitoring** node.
9. Expand a **MonitorConfiguration** node; for example, **MonitorConfiguration.diameterGatewayLatency**.
10. Expand **Attributes**.
11. Verify that the values that you specified in step 2 appears.

> **Note:**
>
> The attributes displayed here are *read-only*. You can update these attributes by editing the *ECE_home***/config/monitor-configuration.xml** file.

Subscribing Notifications

To subscribe for receiving monitoring notifications:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Monitoring** node.
3. Expand the **Notifier** node.
4. Expand **Attributes**.
5. Click **Notifications**.

   The monitoring notifications published to the JMX notification queue appears.
6. Click **Subscribe**.

Reading Log Files

By default, ECE stores log files in the *ECE_home***/logs** directory, but you can configure a new log location. See "Configuring Log Location".

Log file names use the format *node_name***.log**. ECE error messages use this syntax:

```
date_and_time log_level -
error_code - request_ID - customer_ID - message_1 message_2
```

For example:

```
2012-09-11 22:27:09.565 PDT DEBUG -
324984520132531235 - 49de072b-33ae-4f6c-816d-35c25b9ade78 - Cust#6500000587 -
 Successfully retrieved tariffPolicy from customer; candidate balance ids ::
 [Bal#6500000587\]
```

Configuring Log Location

To configure a log location:

1. On the driver machine, open the *ECE_home*/**config/ece.properties** file.
2. Search for and uncomment the following system property:

   ```
   #logDir =
   ```

3. Specify the path to the new log location:

   ```
   logDir = ECE_log_path
   ```

   where *ECE_log_path* is the absolute path to the directory that you want to use as your log location.

   For example:

   ```
   logDir = ECE_home/ece/logs
   ```

4. Save and close the file.

Setting Log Levels

To set log levels:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **ECE Logging**.
4. Expand **Operations**.
5. Set the log level of the appropriate ECE modules. You can set log levels for a module by name or for a set of modules by function (for example, modules used for rerating).
   - To set the log level by ECE module name:
     a. Click the **getLoggerLevels** operation button.
     b. From the list, identify the ECE module relevant to your debugging scenario.
     c. In the input argument of the **setLogLevel** operation, for each parameter (for example, **p1**), replace **String** with the name of the ECE module.

        For example, if you are debugging a problem with using the simulator testing tool, you might set a log level of DEBUG for these ECE modules:

        ```
        oracle.communication.brm.charging.appconfiguration
        oracle.communication.brm.charging.brs
        oracle.communication.brm.charging.tools.simulator
        ```

   - To set the log level by ECE functional domain:
     a. Click the **getFunctionalDomains** operation button.
     b. From the list, identify the ECE functional domain relevant to your debugging scenario.

c. In the input argument of the **setLogLevelForFunctionalDomain** operation for the first parameter (**p1**), replace **String** with the name of the ECE functional domain by pasting the name you copied in the previous step.

   Enter the name of the functional domain exactly as it appears in the list of the **getFunctionalDomains** operation.

d. In the input argument of the **setLogLevelForFunctionalDomain** operation for the second parameter (**p2**), replace String with the log level you want to set for the ECE modules associated with the ECE functional domain.

Configuring the Charging-Server Health Threshold

You can configure a charging-server health threshold so that you are alerted when charging server node failures threaten the ability of your system to handle usage requests. A charging-server health threshold is the minimum number of charging server nodes needed for your customer base. If the number of charging server nodes running on your system goes below the threshold, ECE stops processing usage requests and issues a **SystemHealthException**. ECE continues to process update, management, query, top-up, debit, and refund requests.

When setting a charging-server health threshold, note the following points:

- If a threshold is *N*, you need to run at least *N*+ *1* nodes to have uninterrupted usage processing during a rolling upgrade.
- Configure a minimum of two charging server nodes per machine.

To configure a charging-server health threshold:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **chargingServer**.
4. Expand **Attributes**
5. Set the **degradedModeThreshold** attribute to the minimum number of charging server nodes needed for your customer base (the number that can handle the normal expected throughput for your system). The default is **0**.
6. Save your changes.

Checking If Nodes are Started or Stopped

You can use Elastic Charging Controller (ECC) to find out which nodes are started and stopped.

1. On the driver machine, path to the **/oceceserver/bin** directory.
2. Start ECC

   ```
   ./ecc
   ```

3. Run the **status** command.

   ```
   status
   ```

Checking the Health Status of Charging Server Nodes

To check the ongoing health of ECE charging-server nodes:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **ChargingClient**.
4. Expand **BatchRequestService**.
5. Expand **Attributes**.
6. Check the value of the **SystemHealth** attribute:
   - **HEALTHY**: Charging server nodes are functioning.
   - **DEGRADED**: Charging server nodes are not available.

Configuring KeepAlive for EM Gateway

BRM Connection Manager (CM) and External Manager (EM Gateway) use a pool of connections to send/receive requests. When there were no requests exchanged between CM and EM Gateway for a defined period of time, the idle connections are closed by the firewall. To prevent this, the KeepAlive option is now enabled on the listening sockets. This allows EM Gateway to use the operating system's KeepAlive settings.

> **Note:**
>
> Before starting EM Gateway, ensure that the KeepAlive interval (**tcp_keepalive_interval**) configured for the operating system does not exceed the idle connection timeout configured in the firewall.
>
> By default, the KeepAlive interval for the operating system is set to 7200 seconds. You must reduce this interval so that it does not exceed the idle connection timeout. See your operating system documentation for information on reducing the KeepAlive interval.

By default, EM Gateway is enabled to use the operating system's KeepAlive settings. You can prevent EM Gateway from using the operating system's KeepAlive settings by setting the **socketKeepAlive** entry in the **emGatewayConfigurations.***Instance_Name* (where *Instance_Name* is the name of the EM Gateway instance; for example, emGateway1) section of the *ECE_home***/config/management/charging-settings.xml** file to **0**.

62
Monitoring ECE Components

Learn how to monitor system processes, such as memory and thread usage, of your Oracle Communications Elastic Charging Engine (ECE) components.

Topics in this document:

About Monitoring ECE Components

You can set up your system to monitor ECE components. When configured to do so, each ECE component exposes a REST endpoint that exposes JVM, Coherence, and application metrics from a single endpoint in an OpenMetrics exposition format. You can then use an external centralized metric service, such as Prometheus, to scrape the ECE metrics and store them for analysis and monitoring.

ECE exposes metric data for the following components by default:

- ECE Server
- BRM Gateway
- Customer Updater
- Diameter Gateway
- EM Gateway
- HTTP Gateway
- CDR Formatter
- Pricing Updater
- Radius Gateway
- Rated Event Formatter

Setting up monitoring of these ECE components involves the following high-level tasks:

1. Configuring ECE to scrape JVM, Coherence, and application metrics and expose them through a Micrometer Prometheus endpoint. See "Scraping and Exposing Metrics for ECE".

   The ECE metric data will be exposed through the following endpoint: **http://localhost:19612/metrics**.
2. Setting up a centralized metrics service, such as Prometheus, to scrape metrics from the Micrometer Prometheus endpoint.
3. Setting up a visualization tool, such as Grafana, to display your ECE metric data in a graphical format.

Scraping and Exposing Metrics for ECE

To configure ECE to scrape and expose JVM, Coherence, and application metrics for each ECE node in your system:

1. Open the *ECE_home*/**config/eceTopology.conf** file in a text editor.
2. Add the following information for each ECE node in your system:
   - **metricsPort**: Set this to a non-null port where the metrics will be exposed. The port number must be unique for each host.
   - **isMetricsEnabled**: Set this to **true** to enable monitoring of this node.
3. Save and close the file.
4. Perform a rolling upgrade of the ECE components.

Example 62-1 Exposing Metrics for All ECE Components

This shows sample **eceTopology.conf** entries for exposing the metrics of all ECE nodes except **httpGateway1**, **cdrGateway1**, and **cdrFormatter1**:

```
#node-name                    |role                          |host
name (no spaces!) |host ip|JMX port |start CohMgt |JVM Tuning File|
metricsPort |isMetricsEnabled
ecs1                          |server                        |
localhost              |       |9999     |true         |
defaultTuningProfile|22000  |true
customerUpdater1              |customerUpdater               |
localhost              |       |9996     |false        |
defaultTuningProfile|22004  |true
pricingUpdater                |pricingUpdater                |
localhost              |       |9995     |false        |
defaultTuningProfile|22005  |true
brmGateway                    |brmGateway                    |
localhost              |       |9994     |false        |
defaultTuningProfile|22006  |true
emGateway1                    |emGateway                     |
localhost              |       |9993     |false        |
defaultTuningProfile|22007  |true
ratedEventFormatter1          |ratedEventFormatter           |
localhost              |       |9992     |false        |
defaultTuningProfile|22008  |true
cdrFormatter1                 |cdrFormatter                  |
localhost              |       |19982    |false        |
defaultTuningProfile|       |false
cdrGateway1                   |cdrGateway                    |
localhost              |       |         |false        |
defaultTuningProfile|       |false
diameterGateway1              |diameterGateway               |
localhost              |       |9991     |false        |
defaultTuningProfile|22009  |true
radiusGateway1                |radiusGateway                 |
localhost              |       |9990     |false        |
defaultTuningProfile|22010  |true
httpGateway1                  |httpGateway                   |
```

```
localhost              |       |         |false        |
defaultTuningProfile|       |false
```

ECE Metrics

ECE collects metrics in the following groups to produce data for monitoring your ECE components:

- JVM Metrics
- BRS Metrics
- Coherence Cache Metrics
- Coherence Federated Service Metrics
- Coherence Service Metrics
- Session Metrics
- Rated Events Metrics
- CDR Formatter Metrics

JVM Metrics

The JVM Metrics group contains standard metrics about the central processing unit (CPU) and memory utilization of JVMs, which are members of the ECE grid. Table 62-1 lists the metrics in this group.

Table 62-1 JVM Metrics

Metric Name	Type	Description
jvm_memory_bytes_init	Gauge	Contains the initial size, in bytes, for the Java heap and non-heap memory.
jvm_memory_bytes_committed	Gauge	Contains the committed size, in bytes, for the Java heap and non-heap memory.
jvm_memory_bytes_used	Gauge	Contains the amount of Java heap and non-heap memory, in bytes, that are in use.
jvm_memory_bytes_max	Gauge	Contains the maximum size, in bytes, for the Java heap and non-heap memory.
jvm_memory_pool_bytes_init	Gauge	Contains the initial size, in bytes, of the following JVM memory pools: **G1 Survivor Space**, **G1 Old Gen**, and **G1 Survivor Space**.
jvm_memory_pool_bytes_committed	Gauge	Contains the committed size, in bytes, of the following JVM memory pools: **G1 Survivor Space**, **G1 Old Gen**, and **G1 Survivor Space**.
jvm_memory_pool_bytes_used	Gauge	Contains the amount of Java memory space, in bytes, is in use by the following JVM memory pools: **G1 Survivor Space**, **G1 Old Gen**, and **G1 Survivor Space**.
jvm_buffer_count_buffers	Gauge	Contains the estimated number of mapped and direct buffers in the JVM memory pool.

Table 62-1 (Cont.) JVM Metrics

Metric Name	Type	Description
jvm_buffer_total_capacity_bytes	Gauge	Contains the estimated total capacity, in bytes, of the mapped and direct buffers in the JVM memory pool.
process_cpu_usage	Gauge	Contains the CPU usage information (in percentage) for each ECE component on the server. This data is collected from the corresponding MBean attributes by JVMs.
process_files_open_files	Gauge	Contains the total number of file-descriptors currently available for an ECE component and the descriptors that are in use for that ECE component.
coherence_os_system_cpu_load	Gauge	Contains the CPU load information (in percentage) for each system in the cluster. These statistics are based on the average data collected from all the ECE grid members running on a server.
system_load_average_1m	Gauge	Contains the system load average (the number of items waiting in the CPU run-queue) information for each machine in the cluster. These statistics are based on the average data collected from all the ECE grid members running on a server.
coherence_os_free_swap_space_size	Gauge	Contains system swap usage information (by default in megabytes) for each system in the cluster. These statistics are based on the average data collected from all the ECE grid members running on a server.

BRS Metrics

The BRS Metrics group contains the metrics for tracking throughput and latency of the charging clients that use batch request service (BRS). Table 62-2 lists the metrics in this group.

Table 62-2 ECE BRS Metrics

Metric Name	Metric Type	Description
ece_brs_task_processed	Counter	Tracks the total number of requests that have been accepted, processed, timed out, or rejected by the ECE component. You can use this to track the approximate processing rate over time, aggregate over all client applications, and so on.
ece_brs_task.pending_count	Gauge	Contains the number of requests that are pending by the ECE component.

Table 62-2 (Cont.) ECE BRS Metrics

Metric Name	Metric Type	Description
ece.brs.current.latency.by.type	Gauge	Tracks the latency of a charging client for each charging operation type in the current query interval. This metric provides the latency information for the following operation types: **Initiate**, **Update**, **Terminate**, **Cancel**, **Price_Enquiry**, **Balance_Query**, **Debit_Amount**, **Debit_Unit**, **Refund_Amount**, and **Refund_Unit**.
ece.brs.current.latency	Gauge	Tracks the current operation latency for a charging client in the current scrape interval. This metric contains the BRS statistics tracked using the **charging.brsConfigurations** MBean attributes. This configuration tracks maximum and average latency for an operation type since the last query. The maximum window size for the collecting this data is 30 seconds, so the query has to be run within every 30 seconds. This metric provides the latency information for the following operation types: **Initiate**, **Update**, **Terminate**, **Cancel**, **Price_Enquiry**, **Balance_Query**, **Debit_Amount**, **Debit_Unit**, **Refund_Amount**, **Refund_Unit**, and **Spending_Limit_Report**.

Coherence Cache Metrics

The Coherence Cache Metrics group provides operational and performance statistics for a cache. You can use this metric to track the overall growth rate of certain caches along with other metrics. For more information about the Coherence cache metrics, see "Oracle Coherence MBeans Reference" in *Oracle Fusion Middleware Managing Oracle Coherence*.

Table 62-3 describes the metrics in this group.

Table 62-3 Coherence Cache Metrics

Metric Name	Metric Type	Description
coherence_cache_entries	Integer	Contains the total number of entries present in an ECE cache on an ECE node at the time the query is run.
coherence_cache_total_gets	Long	Contains the total number of get operations performed on an ECE cache since the statistics were last reset.

Table 62-3 (Cont.) Coherence Cache Metrics

Metric Name	Metric Type	Description
coherence_cache_misses	Long	Contains the rough number of total misses performed on an ECE cache since the statistics were last reset.
coherence_cache_total_puts	Long	Contains the total number of put operations made to an ECE cache since the statistics were last reset.
coherence_cache_store_failures	Long	Contains the total number of cache store failures for load, store, and erase operations, since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_read_millis	Long	Contains the cumulative time, in milliseconds, spent on load operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_read_seconds	Long	Contains the cumulative time, in seconds, spent on load operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_reads	Long	Contains the total number of load operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_writes	Long	Contains the total number of store and erase operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_write_millis	Long	Contains the cumulative time, in milliseconds, spent on store and erase operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_store_write_seconds	Long	Contains the cumulative time, in seconds, spent on store and erase operations since the statistics were last reset. The value is **-1** if the cache store type is NONE.
coherence_cache_size	Gauge	Contains the total size of an ECE cache (by default in megabytes).

Coherence Federated Service Metrics

The Coherence Federated Service Metrics group contains metrics for ECE federated caches when ECE persistence is disabled. The metrics in this group provide information regarding the volume of data transferred, the number of objects transferred, and so on. You can use this metric to monitor the data transferred from the primary production system to the remote or backup systems. This data is typically used for disaster recovery where the Oracle NoSQL database is used for storing rated events. For more information about the Coherence federated metrics, see "Oracle Coherence MBeans Reference" in *Oracle Fusion Middleware Managing Oracle Coherence*.

Table 62-4 lists the metrics in this group.

Table 62-4 Coherence Federated Service Metrics

Metric Name	Type	Description
coherence_federated_service_bandwidth	Gauge	Tracks the current or maximum bandwidth used to transfer data from ECE charging nodes to a secondary ECE cluster.
coherence_federated_service_rate	Gauge	Tracks the approximate rate of data transfer in bytes and number of messages sent. This metric uses the **Coherence** MBean attributes for tracking data.
coherence_federated_service_replicate_millis	Gauge	Contains the cache replication latency, in milliseconds, for initial data replication. The type can be: **total** (total time taken for replication) or **estimate_ttc** (estimated time to complete).
coherence_federated_service_replicate_seconds	Gauge	Contains the cache replication latency, in seconds, for initial data replication. The type can be: **total** (total time taken for replication) or **estimate_ttc** (estimated time to complete).
coherence_federated_service_replicate_percent	Gauge	Contains the percentage of cache replication completed for a service.
coherence_federated_service_status	Gauge	Contains the state or status of the service that is on federation. The state of a service can be: **1** (Initial), **2** (Idle), **3** Ready, **4** (Sending), **5** (Connecting), **6** (Connect_Wait), **7** (Stopped), **8** (Paused), **9** (Error), **10** (Yielding), **11** (Backlog_Excessive), **12** (Backlog_Normal), and **13** (Disconnected). The status of a service can be: **1** (OK), **2** (Warning), and **3** (Error).
coherence_federated_service_time_millis	Gauge	Contains the cache replication latency, in milliseconds, for data replicated to the remote cache. The type can be: **apply**, **backlog_delay**, and **round_trip**. These are the 90th percentile latency times.
coherence_federated_service_time_seconds	Gauge	Contains the cache replication latency, in seconds, for data replicated to the remote cache. The type can be: **apply** and **backlog_delay**. These are the 90th percentile latency times.

Table 62-4 (Cont.) Coherence Federated Service Metrics

Metric Name	Type	Description
coherence_federated_service_total	Gauge	Contains the total number of bytes, cache entries, and messages that are replicated to the remote cache. The entity type can be: **records** (total number of journal records), **bytes** (total number of bytes sent), **entries** (total number of cache entries sent), **message** (total number of messages sent), **response** (total number of message responses received). The status can be: **sent** (entity shipped to remote cluster), **unacked** (messages sent without acknowledgment), and **error** (messages failed).

Coherence Service Metrics

The Coherence Service Metrics group contains metrics for the Oracle Coherence cache services. You can use this data to monitor the server latency and load (per node) and the backlogs which may accumulate on the ECE nodes. For more information about the Coherence service metrics, see "Oracle Coherence MBeans Reference" in *Oracle Fusion Middleware Managing Oracle Coherence*.

Table 62-5 lists the metrics in this group.

Table 62-5 Coherence Cache Service Metrics

Metric Name	Type	Description
coherence_service_avg_thread_count	Float	Contains the average active thread count as in the service thread pool.
coherence_service_endangered_partitions	Integer	Contains the total number of partitions that are currently not backed up. The metric value is 0 if the partition is not endangered, and a number greater than zero if any node has failed.
coherence_service_ha_status	String	Contains the high-availability status for this service. The values are ENDANGERED, NODE-SAFE,MACHINE-SAFE, and N/A.
coherence_service_request_avg_duration	Float	Contains the average duration, in milliseconds, of an individual request that was issued by the service since the last time the statistics were reset.
coherence_service_request_count	Long	Contains the total number of synchronous requests run by the service since the last reset.
coherence_service_request_max_duration	Long	Contains the maximum duration, in milliseconds, of a server-side request since the last reset.

Table 62-5 (Cont.) Coherence Cache Service Metrics

Metric Name	Type	Description
coherence_service_request_pending_count	Long	Contains the total number of requests currently pending for a service. A large number of pending tasks may indicate a performance or capacity problem.
coherence_service_request_pending_duration	Long	Contains the duration of the request, in milliseconds, pending in a service.
coherence_service_task_avg_duration	Float	Contains the average server-side task latency in milliseconds since the last reset.
coherence_service_task_backlog	Integer	Contains the current server-side task backlog for each service. A large backlog is indicative of a performance or capacity problem.
coherence_service_task_count	Long	Contains the total number of tasks processed by a service since the last reset.
coherence_service_task_max_backlog	Integer	Contains the maximum size of the backlog queue that holds tasks scheduled to be executed by a service thread.
coherence_service_unbalanced_partitions	Integer	Contains the total number of primary and backup partitions that remain to be transferred until the partition distribution across the storage enabled service members is fully balanced. Typically, unbalanced partitions can occur temporarily during rebalancing operations when the nodes are added or removed.

Session Metrics

The Session Metrics group contains metrics on ECE server sessions. Table 62-6 lists the metrics in this group.

Table 62-6 Session Metrics

Metric Name	Type	Description
ece_session_metrics	Counter	Contains the total number of sessions opened or closed by rating group, node, or cluster.

Rated Events Metrics

The Rated Events Metrics group contains metrics on rated events processed by ECE server sessions. Table 62-7 lists the metrics in this group.

Table 62-7 Rated Events Metrics

Metric Name	Type	Description
ece_rated_events_formatted	Counter	Contains the number of successful or failed formatted rated events per RatedEventFormatter worker thread upon each formatting job operation from NoSQL or the Oracle database.
ece_rated_events_cached	Counter	Contains the total number of rated events cached by each ECE node.
ece_rated_events_inserted	Counter	Contains the total number of rated events that were successfully inserted into the cache.
ece_rated_events_insert_failed	Counter	Contains the total number of rated events that failed to be inserted into the cache.
ece_rated_events_purged	Counter	Contains the total number of rated events that were purged.
ece_requests_by_result_code	Counter	Tracks the total requests processed by using the result code.

CDR Formatter Metrics

The CDR Formatter Metrics group contains the metrics for tracking Charging Function (CHF) records. Table 62-8 lists the metrics in this group.

Table 62-8 CDR Formatter Metrics

Metric Name	Metric Type	Description
ece_chf_records_processed	Counter	Tracks the total number of CHF records that have been processed by the CDR formatter.
ece_chf_records_purged	Counter	Tracks the total number of CHF records that have been purged by the CDR formatter.
ece_chf_records_loaded	Counter	Tracks the total number of CHF records that have been loaded by the CDR formatter.

63

Configuring Subscriber-Based Tracing and Logging

Learn how to configure subscriber-based tracing and logging in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

- About Subscriber-Based Tracing and Logging
- Enabling or Updating Subscriber-Based Tracing and Logging
- Disabling Subscriber-Based Tracing and Logging
- Enabling or Updating Subscriber-Based Tracing and Logging for Selective Functions
- Collecting Subscriber-Based Log Files

See also "Reading Log Files" and "Collecting Log Files for Sending to Oracle Technical Support".

About Subscriber-Based Tracing and Logging

You can selectively trace a particular subscriber's session based on the subscriber ID. You can specify a list of subscriber IDs to be traced. ECE then generates log files for the listed subscribers for each session. If a particular subscriber has multiple sessions, separate log files are generated for each session.

The trace file names are unique and are in the format of *node_name*.*subscriber_ID*.*session_ID*.**log** file. For example, **ecs1.SUBSCRIBER1.SESSION1.log**.

> **Note:**
>
> By default, ECE can trace and log a maximum of 100 subscriber IDs and 24 session IDs per subscriber at a time. However, if required, you can configure these parameters.
>
> ECE does not archive or remove the log files that are generated. Make sure that you remove or archive the log files, periodically, to avoid running out of disk space.

To enable subscriber-based tracing, a new file named **subscriber-trace.xml**, is included in the *ECE_home*/**config** directory. You can update this file to add a list of subscriber IDs whose sessions need to be traced. See "Enabling or Updating Subscriber-Based Tracing and Logging" for more information.

You can also specify if you want to trace and log selective functions, such as alterations (discounts), charges, and distributions (charge sharing), for each subscriber specified in the

list. To enable subscriber-based tracing and logging for selective functions, see "Enabling or Updating Subscriber-Based Tracing and Logging for Selective Functions".

By default, the session trace log files are stored in the *ECE_home*/**logs** directory. However, you can modify the location of the session trace files by updating the **subscriberTraceLogDir** parameter in the *ECE_home*/**config/ece.properties** file.

> **Note:**
>
> By default, the log level is set to **DEBUG** (recommended log level) to generate logs with detailed information. However, you can change the log levels by updating the **logLevel** parameter in the *ECE_home*/**config/subscriber-trace.xml** file.

Enabling or Updating Subscriber-Based Tracing and Logging

To enable or update subscriber-based tracing and logging:

1. Open the *ECE_home*/**config/subscriber-trace.xml** file.
2. Search for the **subscriberTraceConfig** section.
3. Update the values for the following attributes if required:
   - **logMaxSubscribers**: Specify the maximum number of subscribers for whom you want to enable tracing. The default value is **100**.
   - **logMaxSubscribersessions**: Specify the maximum number of sessions for which the logs need to be generated per subscriber. The default value is **24**.
   - **logExpiryWaitTime**: Specify how long to wait, in seconds, before the logging session expires. The default value is **1**.
   - **logCleanupInterval**: Specify the interval time, in seconds, for log cleanup. The default value is **2**.
   - **logLevel**: Specify the log level you want to use for generating logs; for example, DEBUG or ERROR. The default value is **DEBUG**.
4. In the **subscriberList** section, provide the list of subscriber IDs.

   The following is an example of the XML file that includes the **subscriberList** section:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<config>
 <!--
    <subscriberTraceConfig
        logMaxSubscribers = "100"
        logMaxSubscriberSessions = "24"
        logExpiryWaitTime  = "1"
        logCleanupInterval = "2"
        logLevel = "DEBUG"
config-
class="oracle.communication.brm.charging.subscribertrace.configuration.intern
al.SubscriberTraceConfig">
```

```
        <subscriberList config-class="java.util.ArrayList">
            <subscriber id="6500000000" config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal.Sub
scriberImpl"/>
            <subscriber id="6500000001"
config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal.Sub
scriberImpl"/>
        </subscriberList>
</subscriberTraceConfig>
</config>
```

5. Save and close the file.
6. On the machine where you have ECC installed, go to *ECE_home*/**bin**.
7. Start ECC:

   ```
   ./ecc
   ```

8. Run the following command, which deploys the ECE installation onto the server machines:

   ```
   sync
   ```

   The relevant ECE installation files are copied to the server machines in your ECE cluster.
9. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
10. Expand the **ECE Logging** node.
11. Expand **Configuration**.
12. Expand **Operations**.
13. Select **updateSubscriberTraceConfiguration**.
14. Click the **updateSubscriberTraceConfiguration** button.
15. In the editor's MBean hierarchy, expand the **ECE Subscriber Tracing** node.
16. Expand **SubscriberTraceManager**.
17. Expand **Attributes**.
18. Verify that the values that you specified in step 3 appears.

> **Note:**
>
> The attributes displayed here are *read-only*. You can update these attributes by editing the *ECE_home*/**config/subscriber-trace.xml** file.

Disabling Subscriber-Based Tracing and Logging

To disable subscriber-based tracing and logging:

1. Open the *ECE_home*/**config/subscriber-trace.xml** file.
2. Search for the **subscriberTraceConfig** section.
3. Remove all the subscriber IDs from the **subscriberList** section. For example:

```
<subscriber id="6500000001" config-
class="oracle.communication.brm.charging.subscribertrace.configuration.intern
al.SubscriberImpl"/>
```

4. Save and close the file.
5. On the machine on which you have ECC installed, go to *ECE_home*/**bin**.
6. Start ECC:

   ```
   ./ecc
   ```

7. Run the following command, which deploys the ECE installation onto the server machines:

   ```
   sync
   ```

   The relevant ECE installation files are copied to the server machines in your ECE cluster.
8. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
9. Expand the **ECE Logging** node.
10. Expand **Configuration**.
11. Expand **Operations**.
12. Select **updateSubscriberTraceConfiguration**.
13. Click the **updateSubscriberTraceConfiguration** button.

Enabling or Updating Subscriber-Based Tracing and Logging for Selective Functions

> **Note:**
>
> You can enable subscriber-based tracing and logging only for the following selective functions: alterations, charges, and distributions.

To enable or update subscriber-based tracing and logging for selective functions:

1. Open the *ECE_home*/**config/subscriber-trace.xml** file.
2. Go to the **subscriberTraceConfig** section.
3. Update the values for the following attributes, if required:
   - **logMaxSubscribers.** Specify the maximum number of subscribers for whom you want to enable tracing. The default value is 100.
   - **logMaxSubscribersessions.** Specify the maximum number of sessions for which the logs to be generated per subscriber. The default value is 24.
   - **logLevel.** Specify the log level you want to use for generating logs; for example, DEBUG or ERROR. The default value is DEBUG.
4. In the **subscriberList** section, provide the list of subscriber IDs:

The following is an example of the XML file that includes the subscriberList section:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<config>
 <!--
    <subscriberTraceConfig
        logMaxSubscribers = "100"
        logMaxSubscriberSessions = "24"
        logExpiryWaitTime  = "1"
        logCleanupInterval = "2"
        logLevel = "DEBUG"
config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal.Sub
scriberTraceConfig">
        <subscriberList config-class="java.util.ArrayList">
            <subscriber id="811000000000" config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal.Sub
scriberImpl"/>
        </subscriberList>
</subscriberTraceConfig>
</config>
```

5. Search for the **ComponentLoggerList** section.
6. Do the following:
   a. To enable subscriber-based tracing and logging for the alteration function, uncomment the following **componentLogger** blocks in the **componentLoggerList** section:

```
<componentLoggerList config-class="java.util.ArrayList">
           <!-- Add specific logger name and log level to overwrite the out
of the box default logger level
                For example, the following configuration overwrite the logger
named "oracle.communication.brm.charging.subscribertrace.configuration"
                to have log level "ALL" and the logger named
"oracle.communication.brm.charging.ratedevent.publisher to have"
                log level "OFF".
                The logger level can only be "ALL", "DEBUG", "ERROR",
"FATAL", "INFO", "OFF", "TRACE", "WARN"
                -->
            <componentLogger
                    loggerName="ALL"
                    loggerLevel="ERROR"
                     config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal
.ComponentLoggerImpl"/>
          <componentLogger
                  loggerName="oracle.communication.brm.charging.rating.alterat
ion"
                  loggerLevel="DEBUG"

                  config-
class="oracle.communication.brm.charging.subscribertrace.configuration.internal
.ComponentLoggerImpl"/>
</componentLoggerList>
```

   b. To enable subscriber-based tracing and logging for the charging function, uncomment the following **componentLogger** blocks in the **componentLoggerList** section:

```
<componentLoggerList config-class="java.util.ArrayList">
           <!-- Add specific logger name and log level to overwrite the out
```

```
of the box default logger level
                For example, the following configuration overwrite the
logger named
"oracle.communication.brm.charging.subscribertrace.configuration"
                to have log level "ALL" and the logger named
"oracle.communication.brm.charging.ratedevent.publisher to have"
                log level "OFF".
                The logger level can only be "ALL", "DEBUG", "ERROR",
"FATAL", "INFO", "OFF", "TRACE", "WARN"
                -->
            <componentLogger
                    loggerName="ALL"
                    loggerLevel="ERROR"
                     config-
class="oracle.communication.brm.charging.subscribertrace.configuration.in
ternal.ComponentLoggerImpl"/>
            <componentLogger
                     loggerName="oracle.communication.brm.charging.rating
.charge"
                     loggerLevel="DEBUG"
                     config-
class="oracle.communication.brm.charging.subscribertrace.configuration.in
ternal.ComponentLoggerImpl"/>
</componentLoggerList>
```

c. To enable subscriber-based tracing and logging for the distribution (charge sharing) function, uncomment the following **componentLogger** blocks in the **componentLoggerList** section:

```
<componentLoggerList config-class="java.util.ArrayList">
            <!-- Add specific logger name and log level to overwrite the
out of the box default logger level
                For example, the following configuration overwrite the
logger named
"oracle.communication.brm.charging.subscribertrace.configuration"
                to have log level "ALL" and the logger named
"oracle.communication.brm.charging.ratedevent.publisher to have"
                log level "OFF".
                The logger level can only be "ALL", "DEBUG", "ERROR",
"FATAL", "INFO", "OFF", "TRACE", "WARN"
                -->
            <componentLogger
                    loggerName="ALL"
                    loggerLevel="ERROR"
                     config-
class="oracle.communication.brm.charging.subscribertrace.configuration.in
ternal.ComponentLoggerImpl"/>
            <componentLogger
                    loggerName="oracle.communication.brm.charging.rating.
distribution"
                    loggerLevel="DEBUG"
                    config-
class="oracle.communication.brm.charging.subscribertrace.configuration.in
ternal.ComponentLoggerImpl"/>
</componentLoggerList>
```

7. Save and close the file.
8. On the machine where you have ECC installed, go to *ECE_home*/**bin**.
9. Start ECC:

```
./ecc
```

10. Run the following command, which deploys the ECE installation onto the server machines:

    ```
    sync
    ```

    The **sync** command copies the relevant files of the ECE installation onto the server machines in the ECE cluster.

11. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
12. Expand the **ECE Logging** node.
13. Expand **Configuration**.
14. Expand **Operations**.
15. Select **updateSubscriberTraceConfiguration**.
16. Click the **updateSubscriberTraceConfiguration** button.
17. In the editor's MBean hierarchy, expand the **ECE Subscriber Tracing** node.
18. Expand **SubscriberTraceManager**.
19. Expand **Attributes**.
20. Verify that the values that you specified in step 3 appears.

> **Note:**
>
> The attributes displayed here are READ ONLY. You can update these attributes by editing the *ECE_home*/**config/subscriber-trace.xml** file.

Collecting Subscriber-Based Log Files

You can use the **infoCollector** command to collect subscriber-based tracing log files from your ECE system.

To collect subscriber-based log files, run the **infocollector** command with the following parameters:

```
infoCollector [-td dir] [-t "SUBSCRIPTION|SUBSCRIPTION.SESSION IDENTIFIER", "..."]
```

where:

- **-td** *dir* collects all trace files from the provided directory or location. For example:

  ```
  infoCollector -td ECE_home/trace01
  ```

- **-t** *"SUBSCRIPTION|SUBSCRIPTION.SESSION IDENTIFIER", "..."* adds all trace files to the collection that match the subscription session identifiers. For example:

  ```
  infoCollector -t "0048100700.Session01", "0048100702"
  ```

The **infoCollector** command does not collect files from systems on which ECC is running. For example, it does not collect files from the network mediation system.

64

Managing Nodes and Clusters in ECE

Learn how to manage nodes and clusters in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

Adding Nodes

Use the Elastic Charging Controller (ECC) **addNode** command to add nodes. The **addNode** command adds an entry for the node in the topology file.

1. On the driver machine, go to the *ECE_home*/**bin** directory.
2. Start Elastic Charging Controller (ECC).

   ```
   ./ecc
   ```

3. Run the ECC **addNode** command.

   ```
   addNode node_name role host_name host_ip JMX_port start_CohMgt JVM_tuning_file
   ```

   where:

   - *start_CohMgt* specifies if the node is a JMX-management-enabled node, which runs a driver machine. Enter **true** or **false**.
   - *JVM_tuning_file* is the JVM tuning file that contains the tuning profile for the node. This file specifies the number of threads, memory, and heap size. Create this file in the *ECE_home*/**config** directory by copying, renaming, and editing the **defaultTuningProfile.properties** file. See "Configuring JVM Tuning Parameters".
4. Run the ECC **start** command to start the new node.

Removing Nodes

Use Elastic Charging Controller (ECC) **removeNode** command to remove nodes. The **removeNode** command removes the entry for the node from the topology file.

1. On the driver machine, path to the **/oceceserver/bin** directory.
2. Start Elastic Charging Controller (ECC).

```
./ecc
```

3. Run the ECC **stop** command to stop the node.
4. Run the ECC **removeNode** command.

Configuring Cluster Quorum Policy

The cluster quorum policy specifies the minimum number of cluster members that must remain in the cluster, without causing data loss, when the cluster service is terminating suspect members. This ensures that a cluster always contains a sufficient number of cluster members to operate successfully.

A member is considered suspect if it does not respond to network communications and is in imminent danger of being disconnected from the cluster.

You specify the required minimum by setting the **timeout-survivor-quorum** value, which can be calculated using this formula:

timeout-survivor-quorum = (*NumberofHosts* **– 1**) ***** *NumberECSperHost*

where:

- *NumberofHosts* is the number of ECE hosts in your Coherence cluster.
- *NumberECSperHost* is the number of ECE Charging Servers (ECSs) per host.

For example, if your ECE system contains 8 ECE VM hosts with 4 ECSs per VM host, you would set **timeout-survivor-quorum** to 28 [that is, (8 – 1) * 4)].

To configure the cluster quorum policy for ECE:

1. Open the *ECE_home***/config/charging-coherence-override-***environment***.xml** file in a text editor, where *environment* is **dev** for your test or development servers, or **prod** for your production servers.
2. Set the **<timeout-survivor-quorum>** element to the minimum number of cluster members in a cluster when the cluster service is terminating suspect members:

```
<cluster-quorum-policy>
   <timeout-survivor-quorum>value</timeout-survivor-quorum>
</cluster-quorum-policy>
```

3. Save and close the file.

Adding Diameter Gateway Nodes for Online Charging

The ECE installer adds a single instance of a Diameter Gateway node (**diameterGateway1**) to your topology. By default, this single node listens to all network interfaces for Diameter messages.

When adding Diameter Gateway nodes:

- In an ECE production environment, use two Diameter Gateway node instances per machine to allow for failover and additional nodes as needed to handle the expected throughput for your system.
- For a development installation on a single machine, the minimum configuration is three charging server nodes and two Diameter Gateway node instances to allow for failover. Server redundancy is a minimum requirement of ECE installations.

To add Diameter Gateway nodes:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.diameterGatewayConfigurations**.
4. Expand **Operations**.
5. Select **addDiameterGatewayConfiguration**.
6. In the **name** field, enter a name for the Diameter Gateway node.
7. Click the **addDiameterGatewayConfiguration** button.

   You have created a Diameter Gateway node.

   (Optional) To fully configure the Diameter Gateway node now, specify values for all Diameter Gateway node configuration properties. See "Configuring Diameter Gateway Nodes". Alternatively, first create multiple nodes, and later configure them.
8. Open the *ECE_home*/**config/eceTopology.conf** file.
9. Add a row for the Diameter Gateway node instance.
10. For that row, enter the following information:
    - Name of the JVM process for the node instance.

      Enter the name used when the node instance was created in the JMX editor.
    - Role of the JVM process for the node instance.

      Enter the role **diameterGateway**.
    - Host name of the physical server machine on which the node resides.
    - JVM tuning file that contains the tuning profile for the node.
11. Save the file.

Configuring Diameter Gateway Nodes

For each Diameter Gateway node, you must configure connection and performance options.

To configure Diameter Gateway nodes:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.diameterGatewayConfigurations.***Instance_Name*, where *Instance_Name* is the name of the instance to configure.
4. Expand **Attributes**.
5. Specify values for all the attributes required to configure the instance.

   See Table 64-1 for attribute descriptions and default values.
6. Go to the *ECE_home*/**bin** directory.
7. Start the Elastic Charging Controller:

   ```
   ./ecc
   ```

8. Do one of the following:
   - If the Diameter Gateway instance is *not* running, start it.

     The instance reads its configuration information by name at startup.
   - If the Diameter Gateway instance is running, stop and restart it.

Table 64-1 Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
name	"diameterGateway1"	The name of the Diameter Gateway instance. Name Diameter Gateway node instances consistently and uniquely (for example, **diameterGateway1**, **diameterGateway2**, and so on. If you change the name of an existing instance, you must update the name in the *ECE_home***/config/eceTopology.conf** file.
clusterName	"cluster1"	The cluster name of the Diameter Gateway instance. ECE uses both **name** and **clusterName** to identify a Diameter Gateway instance. You must specify a different cluster name if you are configuring Diameter Gateway nodes with the same name. **Note**: Ensure that this cluster name matches the cluster name in the *ECE_home***/config/charging-coherence-override-secure-prod.xml** file.
diameterTrafficHost	""	The network interface (on the physical or virtual host computer running the Diameter Gateway node instance) that the Diameter Gateway node binds to and listens on for Diameter messages. The value can be an IP address or a host name. The value can also be an empty string (the default). In that case, the Diameter Gateway instance listens for Diameter messages on all network interfaces available on the server.
diameterTrafficPort	"3868"	The port (on the physical host computer running the Diameter Gateway node instance) the Diameter Gateway instance listens on to handle Diameter messages. When adding new Diameter Gateway instances, choose a port number that is not used by another application. When multiple Diameter Gateway instances run on the same physical host computer, each instance must use a different port number.

Table 64-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
diameterTrafficHostSctp	""	When SCTP is used, the network interface (on the physical or virtual host computer running the Diameter Gateway node instance) that the Diameter Gateway node binds to and listens on for Diameter messages (sent from Diameter clients). The value can be: • One or more SCTP IP addresses. For multihoming systems, separate multiple IP addresses with a comma (,). For example: `10.240.179.147,10.240.182.149` • One or more host names. • An empty string, which disables the SCTP transport. Your operating system must support SCTP to use this configuration. If not, install the SCTP system package for your operating system version.
sctpMaxInStream	""	The maximum number of incoming streams allowed per association. The valid range is **0** to **65535**.
sctpMaxOutStream	""	The maximum number of outgoing streams allowed per association. The valid range is **0** to **65535**.
sctpReceiveBufferSize	""	The buffer size for receiving data (in bytes). The valid range is **0** to **2147483647**. This attribute determines the SCTP receiver window size. You can increase or decrease the size based on your requirements. For example, you can increase the size for high-volume connections or decrease the size if you want to limit the incoming data backlog.
sctpSendBufferSize	""	The buffer size for sending data (in bytes). The valid range is **0** to **2147483647**. This attribute determines the maximum amount of data that can be sent in a single transaction. You can increase or decrease the size based on your requirements.
originHost	No default value Setting a value for this field is mandatory.	The Origin-Host attribute-value pair (AVP) to be sent in the Diameter request. You assign this unique identifier to your Diameter Gateway server on its host. It can be any string value.
originRealm	No default value Setting a value for this field is mandatory.	The Origin-Realm AVP to be sent by the Diameter Gateway in outgoing Diameter requests. You assign this signaling realm (domain) to your Diameter Gateway server. You must set the same origin realm value for all Diameter Gateway instances in the same ECE topology.

Table 64-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
loopback	"false"	Specifies the loopback setting for performance testing. The valid values are: • **True** specifies that the Diameter Gateway instance does not send the credit-control request to ECE. Instead, the Diameter Gateway instance returns the success result code to the network element. • **False** specifies that the Diameter Gateway instance sends the credit-control request to ECE.
ioThreadPoolSize	"10"	The number of threads used by the network I/O thread pool. The valid values are greater than zero and up to any number the system resources allow.
responseTimeout	"10"	The maximum duration, in seconds, the Diameter Gateway instance waits for a response from the Diameter client. If the Diameter Gateway instance does not receive a response from the Diameter client within the specified duration, the Diameter Gateway instance stops waiting for a response and removes the notification from the JMS queue. The valid values are greater than zero and up to any number the system resources allow.
requestProcessorThreadPoolSize	"10"	The number of threads used by the request-processor thread pool. The request-processor thread pool is a Diameter Gateway thread pool dedicated to processing Diameter requests handed off to it from the I/O thread pool. The valid values are greater than zero and up to any number the system resources allow.
requestProcessorBatchSize	"10"	The batch size of the Diameter requests handed off by the network I/O thread pool to the request-processor thread pool. The valid values are greater than zero and up to any number the system resources allow.
watchDogInterval	"30"	The duration in seconds that the Diameter Gateway instance waits before it issues a Device-Watchdog-Request message (DWR).
notificationThreadPoolSize	"10"	The number of threads used by the Diameter Gateway instance to process notification messages. The valid values are greater than zero and up to any number the system resources allow. Tune this value to the expected workload in the deployed environment.
maxNotificationCommitSize	"100"	The maximum number of dequeued notification messages from the JMS topic that can remain uncommitted. If the number of dequeued notification messages from the JMS topic exceeds this number, the Diameter Gateway instance stops reading messages until the read messages are committed.

Table 64-1 (Cont.) Diameter Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
ccFailover	"FAILOVER_SUPPORTED"	Indicates if the Diameter Gateway instance operates in a cluster that supports failover. The valid values are: • **FAILOVER_SUPPORTED** • **FAILOVER_NOT_SUPPORTED** The value set here is the value the Diameter Gateway instance sends for the CC-Session-Failover AVP in all credit-control answers (CCAs) it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.4
creditControlFailureHandling	"RETRY_AND_TERMINATE"	Indicates how the Diameter client should proceed if a CCA is not received before the Tx timeout. The valid values are: • **TERMINATE** • **CONTINUE** • **RETRY_AND_TERMINATE** The value set here is the value the Diameter Gateway instance sends for the Credit-Control-Failure-Handling AVP in all CCAs it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.14
directDebitingFailureHandling	"TERMINATE_OR_BUFFER"	Indicates how the Diameter client should proceed if a Direct Debit CCA is not received before the Tx timeout. The valid values are: • **TERMINATE_OR_BUFFER** • **CONTINUE** The value set here is the value the Diameter Gateway instance sends for the Direct-Debiting-Failure-Handling AVP in all credit-control answers (CCAs) it produces. For more information, see the Diameter Credit-Control Application standard at: https://tools.ietf.org/html/rfc4006#section-8.15

Adding RADIUS Gateway Nodes

During ECE installation, if you specify that RADIUS Gateway must be started when ECE is started, the ECE installer adds a single RADIUS Gateway node (**radiusGateway1**) to your topology file).

When adding RADIUS Gateway nodes:

- In a production system, use two RADIUS Gateway nodes to allow for failover and additional nodes as needed to handle the expected throughput for your system.

- For a development system, the minimum configuration is two RADIUS Gateway nodes to allow for failover.

To add RADIUS Gateway nodes:

1. Log on to the driver machine.
2. Go to the *ECE_home*/**bin** directory.
3. Start the Elastic Charging Controller:

   ```
   ./ecc
   ```

4. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
5. Expand the **ECE Configuration** node.
6. Expand **charging.radiusGatewayConfigurations**.
7. Expand **Operations**.
8. Select **addRadiusGatewayConfiguration**.
9. In the method's **name** field, enter a name for the RADIUS Gateway node.
10. Click the **addRadiusGatewayConfiguration** button.

    You have created a RADIUS Gateway node.
11. Open the *ECE_home*/**config/eceTopology.conf** file.
12. Add a row for the RADIUS Gateway node instance.
13. For that row, enter the following information:
    - Name of the JVM process for the node instance.

      Enter the name used when the node instance was created in the JMX editor.
    - Role of the JVM process for the node instance.

      Enter the role **radiusGateway**.
    - Host name of the physical server machine on which the node resides.
    - JVM tuning file that contains the tuning profile for the node.
14. Save the file.

Configuring RADIUS Gateway Nodes

You must configure each RADIUS Gateway node to communicate with your network and to perform optimally.

To configure RADIUS Gateway nodes:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.radiusGatewayConfigurations.***Instance_Name*, where *Instance_Name* is the name of the instance to configure.
4. Expand **Attributes**.
5. Specify values for all the attributes required to configure the instance.

See Table 64-2 for attribute descriptions and default values.

6. Change directory to the *ECE_home***/bin** directory.
7. Start the Elastic Charging Controller:

```
./ecc
```

8. Do one of the following:
   - If the RADIUS Gateway instance is *not* running, start it.

     The instance reads its configuration information by name at startup.
   - If the RADIUS Gateway instance is running, stop and restart it.

Table 64-2 RADIUS Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
name	"radiusGateway1"	The name of the RADIUS Gateway instance. Name RADIUS Gateway node instances consistently and uniquely (for example, **radiusGateway1**, **radiusGateway2**, and so on). If you change the name of an existing instance, you must update the name in the *ECE_home***/config/eceTopology.conf** file.
wallet	"opt/wallet"	The path to the Oracle wallet file containing the SSL trusted certificates and the BRM root key for RADIUS Gateway. When RADIUS Gateway is started, the BRM root key in the Oracle wallet file is stored in memory.
queueSize	"8"	The number of incoming requests that can be simultaneously processed by the RADIUS server.
avpName	"Service-Type"	The name of the attribute value pair (AVP) that is used to determine the service type during authentication. This is used in conjunction with **vendorId**.
vendorId	"0"	The vendor ID of the AVP that you configured to determine the service type. This is used in conjunction with **avpName**.
enableRetransmissionChecks	"true"	Enables or disables duplicate packet detection (enabled by default). RADIUS Gateway uses this feature to identify duplicate requests from RADIUS clients by validating them against the requests stored in the ECE cache. **Note:** You must restart RADIUS Gateway after enabling or disabling the duplicate packet detection feature.
timeToLive	"30000"	The expiration time (in milliseconds) for the RADIUS requests stored in the ECE cache.
radiusTrafficPort	authorization port= "1812" accounting port= "1813"	The number assigned to the port on which RADIUS Gateway listens. Add one **radiusTrafficPort** entry for each port on which you want RADIUS Gateway to listen.

Table 64-2 (Cont.) RADIUS Gateway Node Configuration Attributes and Values

Attribute Name	Default Value	Description
ioThreadPoolSize	"16"	The number of threads that determines the maximum number of simultaneous processes that RADIUS Gateway can handle. You can increase the number of threads to increase the server performance. Many factors impact the number of threads required, such as the cache size of each CPU, memory size, and swap size. Systems can handle as many as eight threads per CPU. On production systems, set these values higher.
sharedSecret	"e59VPnxr1o5+FGW97w/aMA=="	The common password shared between RADIUS Gateway and Network Access Server (NAS). It is used by the RADIUS protocol for security. Each RADIUS Gateway instance must have a unique password in encrypted format.
noOfChallenges	"1"	The maximum number of challenges that can be sent to RADIUS clients when Challenge-Handshake Authentication Protocol (CHAP) is used for authentication. A random number within this value is chosen during authentication to carry out the number of challenges for a given authentication session. If the password is authenticated successfully, the challenge process begins. If any of the challenge responses fail in authentication, an Access-Reject is sent. Upon all successful authentication, an Access-Accept message is sent.
id	"21", "4", and "5"	The unique identifier of the Extensible Authentication Protocol (EAP) type used for authentication.
name	"TTLS" and "MD5"	The EAP type used for authentication. The following types are supported: TTLS and MD5.
priority	"1", "2", and "3"	The priority set for the EAP type. 1 is the highest priority.

65

Configuring ECE Data-Loading Utilities and Data Updaters

Learn how to change the initial configuration of the Oracle Communications Elastic Charging Engine (ECE) data-loading utilities and data updaters after you install ECE.

Topics in this document:

Configuring the configLoader Utility

Use the **configLoader** utility to load mediation specification data required for Diameter Gateway and RADIUS Gateway into ECE.

To configure the **configLoader** utility:

1. Open the *ECE_home***/config/management/migration-configuration.xml** file.
2. Set the **configObjectsDataDirectory** parameter to the path to the mediation specification file that is supplied (as ready-to-use configuration) when you install ECE. The default path is *ECE_home***/sample_data/config_data** directory.

Loading Configuration Data with configLoader

To load configuration data with **configLoader**:

1. Go to the *ECE_home***/bin** directory.
2. Start ECC:

   ```
   ./ecc
   ```

3. If your charging servers are not running, start them:

   ```
   start server
   ```

4. Run the following command:

   ```
   start configLoader
   ```

Configuring the pricingLoader Utility

To configure the **pricingLoader** utility:

1. Open the *ECE_home***/config/management/migration-configuration.xml** file.

2. Set the **pricingDataDirectory** parameter to the path of the directory that contains the pricing data XML data files.

   pricingLoader loads the pricing data in the XML data files published to this directory into ECE caches.

   By default, this parameter is set for loading sample pricing component data from the *ECE_home***/config/management/sample_data/pricing_data** directory.

3. Set the **configObjectsDataDirectory** parameter to the path of the directory that contains the ECE configuration data XML data file.

   pricingLoader loads the configuration data in the XML data files published to this directory into ECE caches.

   By default, this parameter is set for loading sample configuration data from the *ECE_home***/sample_data/config_data** directory.

Loading Pricing Data with pricingLoader

To load sample pricing data with **pricingLoader**:

1. Change directory to the *ECE_home***/bin** directory.
2. Start ECC:

   ```
   ./ecc
   ```

3. If the charging server nodes are not running, start them and load configuration data:

   ```
   start server
   start configLoader
   ```

4. Run the following command, which loads the pricing data:

   ```
   start pricingLoader
   ```

Configuring the customerLoader utility

> **Caution:**
>
> Do *not* run the **customerLoader** utility without the **-incremental** parameter in a production environment.

To configure the **customerLoader** utility:

1. Open the *ECE_home***/config/management/migration-configuration.xml** file.
2. Set the **configObjectsDataDirectory** parameter to the path of the directory that contains the credit limit and product offering XML data files.

   The **customerLoader** utility loads the sample profile data from the XML data files in this directory into the ECE customer cache.

   By default, this parameter is set to *ECE_home***/sample_data/config_data**.

3. Set the **productOfferingCrossRefFilePath** parameter to the path of the directory for the product offering cross-reference XML data file.

The **customerLoader** utility loads the product offering cross-reference data from the XML data files in this directory into the ECE customer cache.

By default, this parameter is set to *ECE_home*/**config/management/sample_data/customer_data**.

4. Set the **customerDataDirectory** parameter to one of the following:
   - To load the sample customer data files in ECE, specify the path of directory for the ECE customer XML data file.

     The **customerLoader** utility loads the customer data in the XML data files published to this directory into the ECE customer cache.

     By default, this parameter is set to *ECE_home*/**config/management/sample_data/customer_data**.
   - To incrementally load the customer data from BRM into ECE, specify the schema from which the data has to be loaded into ECE. The schema and file name associations are found in the **connectionConfigurations** MBean.
5. Set the **customerXmlPattern** parameter to the pattern of the customer XML data file *name* that you want **customerLoader** to load. The files with a name that starts with this pattern are loaded.

   By default, this parameter is set to **customer**.
6. Set the **remoteWmThreads** parameter to the number of parallel work manager threads to be used for the **customerLoader** utility.

   By default, this parameter is set to **1**.
7. Set the **batchSize** parameter to the number of customers to put into the repository in one put operation.

   Use **batchSize** to optimize performance of put operations into the repository. The more customers you insert into the repository in one put operation (rather than inserting each one individually), the better the performance.

   By default, this parameter is set to **5000**.
8. Save and close the file.

Loading Customer Data Incrementally with customerLoader

> **Note:**
>
> Do *not* run the **customerLoader** utility without the **-incremental** parameter in a production environment.

After loading the initial customer data from BRM, you can load customer data incrementally, in batches or in bulk, into ECE by using the **customerLoader** utility. On a system with multiple schemas, run the **customerLoader** utility for each schema.

To load customer data incrementally into ECE:

1. Start ECC:

```
./ecc
```

2. If your charging server nodes are not running, start them and load configuration data and pricing data:

```
start server
start configLoader
start pricingLoader
start CustomerUpdater
```

3. Create prepopulated tables to load only preselected customer data. For more information, see "Loading a Subset of Customer Data".

4. Run the following command, which loads the customer data incrementally:

```
start customerLoader -incremental customer_updater_schema_name
```

where *customer_updater_schema_name* is the schema name specified for Customer Updater in the **connectionconfiguration** section of the *ECE_home***/config/management/charging-settings.xml** file.

For Example:

```
start customerLoader -incremental customerUpdater1
```

Loading Product Cross-Reference Data with customerLoader

Running the **Customer Loader** utility with the **-loadCrossRefData** parameter loads the new or modified set of product cross-reference data from BRM into ECE for the schema that you specify. The **Customer Loader** utility loads the product cross-reference data only for the product offerings that are stored in the ECE cache when the utility is run with the **-loadCrossRefData** parameter.

To load product cross-reference data with the **customerLoader** utility:

1. Start ECC:

```
./ecc
```

2. If your charging server nodes are not running, start them and load configuration data and pricing data:

```
start server
start configLoader
start pricingLoader
start CustomerUpdater
```

3. Run the following command, which loads the cross-reference data from the specified customer updater schema:

```
start customerLoader -loadCrossRefData customer_updater_schema_name
```

where *customer_updater_schema_name* is the schema name specified for Customer Updater in the **connectionconfiguration** section of the *ECE_home***/config/management/charging-settings.xml** file.

For example:

```
start customerLoader -loadCrossRefData customerUpdater1
```

For loading cross-reference data from multiple schemas, run the **customerLoader** utility for each schema; for example, customerUpdater1, customerUpdater2, and so on.

Configuring Customer Updater

To configure Customer Updater:

> **Note:**
>
> For Customer Updater to receive update requests from BRM, the BRM Payload Generator External Module (EM) configuration file must point to the ECE payload configuration file (**payloadconfig_ecc_sync.xml**).

1. To enable Customer Updater to work with prepopulated distribution tables, set **-DpreDistributedWorkItems** to **true** in the JVM tuning file.

   The prepopulated distribution tables specify which customer data to load. Only data for the customers listed in the prepopulated distribution tables is loaded. Customer Updater loads all BRM customer data by default.
2. To enable Customer Updater to continue data extraction even if validation fails, set **-DcontinueCustomerLoaderOnError** to **true** in the JVM tuning file.
3. Open the *ECE_home***/config/management/migration-configuration.xml** file.
4. Set the **remoteWmThreads** parameter to the number of parallel work manager threads to be used for Customer Updater.

   By default, this parameter is set to **1**.
5. Set the **batchSize** parameter to the number of customers to put into the repository in one put operation.

   Use **batchSize** to optimize performance of put operations into the repository. The more customers you insert into the repository in one put operation (rather than inserting each one individually), the better the performance.

   By default, this parameter is set to **5000**.
6. Set the **dbConnections** parameter to the number of parallel database connections to use for initial extraction and load of customer data.

   By default, this parameter is set to **1**.
7. Set the **dbFetchSize** parameter to the number of records that Customer Updater should extract from the BRM database at one time.

   By default this parameter is set to **5000**.
8. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
9. Expand the **ECE Configuration** node.
10. Expand **charging.connectionConfigurations.oracleQueueConnection1**.
11. Expand **Attributes**.
12. Specify values for the following attributes:
    - **jdbcUrl**: Enter the Oracle JDBC URL to use to connect to the BRM database.

      ```
      jdbcUrl="jdbc:oracle:thin@//hostname:port:sid"
      ```

where *hostname* and *port* are the host name and port number for the computer on which the database queue resides, and *sid* is the name of the BRM database service.

- **Password**: Enter the encrypted password for logging on to the computer on which the database queue resides.

  When you install ECE, the password you enter is encrypted and stored in the KeyStore.

- **QueueName**: Enter the name of the database queue that holds the published business events from BRM.

Specifying Retry Count for Customer Updater

You can configure Customer Updater to automatically retry the BRM database connection and restart the process by specifying the number of retries allowed after it fails. This ensures that Customer Updater is started automatically without any manual intervention.

To specify the retry count for Customer Updater:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations.customerUpdater***n*, where *n* is the number that represents the instance; for example, CustomerUpdater2.
4. Expand **Attributes**.
5. Set the **retryCount** attribute to the number of times the database connection can be retried after Customer Updater fails.

   This attribute is applicable only for the Customer Updater startup, restart, or failover.

Loading a Subset of Customer Data

By default, Customer Updater loads all customer data from the BRM database into ECE. However, you can configure Customer Updater to load only a subset of customer data.

To define the subset of data to load:

1. Open the *ECE_home***/bin/preselect_customer.groovy** file.
2. Specify the customer filter criteria to use for selecting which customers to load by updating the **filterCondition** string. Add the customer query criteria to the where clause of the query on the ACCOUNT_T table.

> **Note:**
>
> The filter conditions you specify are added to the SQL query that is used to select the customer account POIDs from the BRM ACCOUNT_T table.

3. Enter the following command:

```
./preselect_customers.sh -u dbUser -p dbPassword -s dbHost -o oracleSID -c dbPort -
n noOfdbConnections -f dbFetchSize
```

where:

- *dbUser* is the name of the BRM database user.
- *dbPassword* is the password of the BRM database user.
- *dbHost* is the host name or IP address of the BRM database user.
- *oracleSID* is the Oracle database alias.
- *dbPort* is the number for the Oracle database port.
- *noOfdbConnections* is the number of connections to use when retrieving BRM customer data from the database. You must set *noOfdbConnections* and the Customer Updater **dbConnections** parameter to the same value.
- *dbFetchSize* is the number of records to fetch from the database at one time. You must set *dbFetchSize* and the Customer Updater **dbFetchSize** parameter to the same value.

The distribution tables that Customer Updater uses to load BRM customers into ECE are created and populated with details of the selected customers.

In a multischema environment, run the **preselect_customer.sh** script in each schema. When you run the script, you specify the database credentials for that schema.

Loading Customer Data Selectively from BRM into ECE

The initial data load includes the customer data only for the product offerings (services and the corresponding pricing data) that are already stored in the ECE cache. If you are migrating pricing data selectively from BRM into PDC, you must load the customer data also selectively from BRM into ECE.

To load customer data selectively from BRM into ECE:

1. Configure **Customer Updater** to run in the **selectiveMigrationMode** mode by setting the **selectiveMigrationMode** attribute in the *ECE_home*/**config/management/migration-configuration.xml** file to **true**.

   See "Configuring Customer Updater to Load Data Selectively".

2. Load the selectively migrated pricing data (including services) from PDC into ECE by running Pricing Updater. See "Starting and Stopping ECE ".
3. Load the corresponding customer data and product cross-reference data from BRM into ECE by running Customer Updater. See "Starting and Stopping ECE ".

Configuring Customer Updater to Load Data Selectively

To configure Customer Updater to load data selectively:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **migration.loader**.

4. Expand **Attributes**.
5. Set the **selectiveMigrationMode** attribute to **true**.

Configuring the Customer Updater Suspense Queue

ECE moves failed data updates from Customer Updater to the suspense queue.

To configure the Customer Updater suspense queue:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations.oracleQueueConnection1**.
4. Expand **Attributes**.
5. Set the **suspenseQueueName** attribute to the name of the queue on the BRM system that holds failed updates from BRM.

Configuring Pricing Updater

To configure the Pricing Updater:

1. Open the *ECE_home*/**config/JMSConfiguration.xml** file.
2. Edit the **PdcEceQueue** section to specify the JMS queue details from which Pricing Updater retrieves the pricing data from PDC.
3. Edit the **PDCResultQueue** section to specify the JMS result queue details to which Pricing Updater publishes the results back to PDC.
4. Save the file.

If you need to change the password values in this file (for example, if ECE must interact with new BRM or PDC machines or if the password used to connect to existing BRM and PDC machines has changed), you must first run the encrypt password utility.

66

Configuring JVM Tuning Parameters

Learn how to tune the JVM parameters in your Oracle Communications Elastic Charging Engine (ECE) system.

Topics in this document:

Configuring JVM Tuning Parameters

Configure JVM tuning parameters for garbage collection and heap size tuning.

Each row in the topology file represents an ECE component that is a running JVM in the cluster. In the topology file, you can specify a JVM tuning profile file for each node defined. This allows you to provide specific tuning settings for each node in the cluster. Multiple nodes can point to the same tuning profile.

To configure JVM tuning parameters:

1. Open the *ECE_home*/**config/defaultTuningProfile.properties** file.

   You can create your own JVM tuning file and save it in this directory. You can name the file what you want.
2. Set the parameters as needed.
3. Save the file.
4. In the topology file, ensure your JVM tuning file is associated with the node to which you want the parameters to apply.

   The JVM tuning file is referenced by name in the *ECE_home*/**config/eceTopology.conf** file.

Configuring Client-Side ECE Request Queues

You can configure the request queues that your charging clients use to submit requests to ECE.

To configure client-side ECE request queues:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **ChargingClient**.
4. Expand **BatchRequestService**.
5. Set the thread pool size, batch time out, and batch size attributes for the request queues.

For descriptions of each attribute, see the documentation for the BRSStatMXBean for **oracle.communication.brm.charging.brs** in *ECE Java API Reference*.

67

Configuring System Overload Protection

Learn how to implement system overload protection in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

Configuring System Overload Protection

System overload causes charging nodes to become stuck or busy. The following scenarios might cause an overloaded system:

- An undersized ECE deployment
- Large batches of offline records
- Bulk customer updates that trigger numerous update requests

When the usage request throughput from ECE is too large for the system to handle, ECE reduces its throughput until it reaches a sustainable error-free level. It informs the client of any submitted requests that are not able to be processed. Update, management, query, top-ups and debit refunds requests are always accepted, even when the system is overloaded.

Overload protection uses thread pools to accept and process requests submitted to the system. Thread pools improve performance when executing large numbers of updates because of the reduced per-update overhead. They also provide a means of bounding and managing the resources, including requests.

The value of the pending count should generally be at least equal to the number of thread counts. Select this value carefully, based on the expected throughput of the ECE instance and the expected latency of each request as revealed by your performance testing results.

To configure system overload protection:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **BatchRequestService**.
4. Expand **Attributes**.
5. Specify values for the overload protection attributes described in Table 67-1.
6. Save your changes.

> **Note:**
>
> When the following MBean attributes are set, they are not persisted. If the client is restarted, attributes are reset to their default values.

Table 67-1 MBean Attributes to Configure Overload Protection

MBean Attribute	Description
AcceptablePendingCount	Number of requests accepted and queued to be processed. ECE rejects the requests if the number of pending requests in the queue exceeds this value. This value should be greater than the value of the **Thread Pending Count** attribute.
AcceptedTaskCount	Number of requests accepted for processing since the start of the ECE instance. This attribute is read-only.
BatchSize	Size of the ECE batch when it is submitted for processing.
BatchTimeOut	Amount of time ECE waits before the batch is submitted for processing, irrespective of how much the batch is filled.
OverloadProtection	Flag that enables overload protection. The default is disabled (**false**).
RejectedTaskCount	Number of requests rejected since the start of the ECE instance. This attribute is read-only.
ThreadPendingCount	Number of requests pending in the queue. This attribute is read-only.

68
Configuring System Currency

Learn how to configure Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) to use a default system currency.

Topics in this document:

Configuring Default System Currency

You can configure ECE to use a default system currency for charging subscribers. During rating, ECE uses the subscriber's primary currency or the secondary currency to charge subscribers. If the currency used in the rate plans does not match the subscriber's primary or secondary currency, ECE uses the default system currency, US dollars.

To configure a default system currency:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes** and select **systemCurrencyNumericCode**.
5. Set the numeric code of the currency for the system.

   For descriptions of each attribute, see the documentation for the BRSStatMXBean for **oracle.communication.brm.charging.brs** in *ECE Java API Reference*.

69

Managing Persisted Data in the Oracle Database

Learn about data persistence and the tasks for managing Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) data stored in an Oracle database.

Topics in this document:

About Persisting Data in the Oracle Database

ECE persists ECE cache data in the Oracle database (referred to as ECE persistence database) to create a permanent backup of the cache data (in-memory data). ECE recovers data automatically from the persistence database in case of a node failure or partition loss.

You enable data persistence and create the schema and tablespaces for storing the ECE data during the ECE installation or upgrade process.

> **Note:**
>
> If you are enabling data persistence, Oracle NoSQL Database is not required. Rated events are directly persisted in the ECE persistence database.

When data persistence is enabled, the parameters for connecting to the persistence database are set by default in the **OraclePersistenceConnectionConfiguration** section of the *ECE_home***/config/management/charging-settings.xml** file (where *ECE_home* is the directory in which ECE is installed). You can reset these parameters based on your business requirements by editing the **charging.connectionConfigurations.***Connection_Name* MBean attributes using a JMX editor, where *Connection_Name* is the name of the persistence database connection such as oraclePersistence1.

When you start the ECE core components, such as Customer Updater, Pricing Updater, and configLoader, the data published from BRM, PDC, and the mediation specification data loaded into the ECE cache are stored in the persistence database by default.

Data that is synchronized from BRM is stored in the persistence database. Other data, such as balance, topup history, recurring bundle history, rated events, and Portal object IDs (POIDs), are also stored in the persistence database.

Active sessions in ECE are not persisted. If ECE restarts or partition loss occurs, active session cache data is lost. The corresponding expired balance reservations are cleaned up so that reserved balances can be used for future usage requests.

Accessing ECE Configuration MBeans

For all configurations, start by accessing the ECE configuration MBeans:

1. Log on to the driver machine.
2. Start the ECE charging servers (if they are not started).
3. Connect to the ECE charging server node enabled for JMX management. This is the charging server node set to **start CohMgt = true** in the *ECE_home*/**config/eceTopology.conf** file, where *ECE_home* is the directory in which ECE is installed.
4. Start a JMX editor that enables you to edit MBean attributes, such as JConsole.
5. In the editor's MBean hierarchy, find the ECE configuration MBeans.

Enabling Multischema Support in the Persistence Database

By default, the ECE persistence database supports only one schema, but you can configure it to support multiple schemas to distribute the database load. When deploying ECE as multischema, the number of schemas in BRM and ECE must be the same.

> **Note:**
>
> ECE migration from a single-schema persistence database to a multischema persistence database is not supported through a rolling upgrade. To migrate data from a single schema persistence database to a multischema persistence database, you must clear all data from the ECE schema, do a cold restart, and then reload the data from BRM and PDC.

To enable multischema support in the persistence database:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Do the following for each secondary schema in your database:
   a. Expand **charging.connectionConfigurations**.

b. Expand **Operations**.

c. Click **addOraclePersistenceConnection**.

d. Set the **name** attribute to **oraclePersistence***N*, where *N* is the number that represents the instance number. For example, oraclePersistence2.

4. Do the following for each schema in your database:

a. Expand **charging.connectionConfigurations.oraclePersistence***N*, where *N* is the number that represents the instance number. For example, oraclePersistence1 or oraclePersistence2.

b. Expand **Attributes**.

c. Set **schemaNumber** to the schema number, such as 1 for the primary schema, 2 for the second schema, and so on.

5. Expand **charging.cachePersistenceConfigurations.***Cluster_Name*, where *ClusterName* is the name of the ECE cluster that you want to update.

6. Expand **Attributes**.

7. Set **persistenceConnectionName** to include the name of each persistence connection, delimited by colons and arranged in ascending schema number order. For example, if your database contains three schemas, you would set it to **oraclePersistence1:oraclePersistence2:oraclePersistence3**.

About Persisting Rated Events and Customer Data

Rated events are loaded directly into the persistence database and initial customer data is loaded into the ECE cache and persisted in the persistence database.

To extract rated events and customer data updates, configure your Rated Event Formatter and Customer Updater instances to connect to the persistence database.

Configuring Rated Event Formatter and Customer Updater to Connect to the Database

To configure the Rated Event Formatter (RE Formatter) and Customer Updater instances to connect to the persistence database for storing the rated events and customer data, you must perform this for each RE Formatter and Customer Updater instance. For more information about configuring the RE Formatter, see "Configuring RE Formatter" in *Loading Rated Events*.

To connect a Rated Event Formatter instance to the persistence database:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".

2. Expand the **ECE Configuration** node.

3. Expand **charging.ratedEventFormatters.***Instance_Name*, where *Instance_Name* is the name of the instance you want to configure, for example, **ratedEventFormatter2**.

4. Expand **Attributes**.

5. Set the **connectionName** attribute to the persistence database connection name, for example, **oraclePersistence1**.

6. Under the ECE Configuration node, expand **charging.CustomerUpdaters.***Instance_Name*, where *Instance_Name* is the name of the instance you want to configure, for example, **customerUpdater2**.
7. Expand **Attributes**.
8. Set the **connectionName** attribute to the persistence database connection name, for example, **oraclePersistence1**.
9. Go to the *ECE_home***/bin** directory.
10. Start ECC:

    ```
    ./ecc
    ```

11. Restart any Rated Event Formatter and Customer Updater instances that you configured. See "Starting ECE".

About Persisting POIDs

For tracking the bill items created in ECE, ECE persists the POID pool received from BRM in the persistence database. Even after ECE restarts, the reserved POID pool is retained in ECE and the POID allocation is continued without interruption.

You configure the frequency at which ECE persists POIDs in the persistence database and the count after which the POIDs are persisted by using the **poidPersistenceSafecount** attribute. By default, this attribute is set to 12000, but you can reset it to meet your business requirements.

To update the count for persisting POIDs:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.itemAssignmentConfig**.
4. Expand **Attributes**.
5. Set the **poidPersistenceSafecount** attribute as needed.

Enabling ECE Cache Override from Original Source

After initial loading of data, when you restart the core components, the data in the persistence database are reloaded into the ECE cache by default. If you want the data to be reloaded directly from BRM, PDC, or any files into the ECE cache, you must configure ECE to override the cache data from the original source before restarting the core components.

You can also reload the customer data incrementally from BRM into ECE if required. For more information, see "Loading Customer Data Incrementally with customerLoader".

To enable ECE cache override from the original source:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.

3. Expand **charging.cachePersistenceConfigurations.***Cluster_Name*, where *Cluster_Name* is the name of the ECE cluster you want to update such as BRM.
4. Expand **Attributes**.
5. Set the following attributes as appropriate:
   - **configLoadFromPersistence**: Set this attribute to **false** to load data from files, BRM, and PDC at the time of restart.

     Setting this attribute to **false** sets the **pricingLoadFromPersistence** and **customerLoadFromPersistence** attributes to **false**.
   - **pricingLoadFromPersistence**: Set this attribute to **false** to load data from PDC at the time of restart.
   - **customerLoadFromPersistence**: Set this attribute to **false** to load data from BRM at the time of restart.

Enabling Partition Recovery

In case of partition loss or a node failure, you can recover the data from lost partitions by enabling partition recovery. However, ECE cannot recover data from persistence if:

- The number of nodes in the ECE system is less than the number of nodes specified in the **DegradedModeThreshold** parameter in the *ECE_home***/config/management/charging-settings.xml** file.
- The status of the ECE system is none of the following:
  - USAGE_PROCESSING
  - INCREMENTAL_CONFIG_DATA_LOADING_IN_PROGRESS
  - INCREMENTAL_PRICING_DATA_LOADING_IN_PROGRESS
  - INCREMENTAL_PRODUCT_OFFERING_CROSS_REF_DATA_LOADING_IN_PROGRESS
  - DEGRADED_MODE
- All of the server nodes or all of the nodes in ECE are shutdown by running the **stop server** or **stop** command.

To enable partition recovery:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.cachePersistenceConfigurations.***Cluster_Name*, where *Cluster_Name* is the name of the ECE cluster you want to update; for example, BRM.
4. Expand **Attributes**.
5. Set the **partitionLossRecoverFromPersistence** to **true** to recover the data from lost partitions.

Enabling Partial Loading of Data

You can enable partial loading of data, such as customer data, into the ECE cache by setting the following for each charging server node in your server:

- The maximum limit, such as **HighUnits**, for loading the data into ECE cache.
- The minimum limit, such as **LowUnits**, for loading or reloading data into ECE cache from the persistence database.

When partial loading is enabled, the initial data load includes only specific data units. When processing usage requests for a customer, if the required data is not available in ECE cache, ECE loads that data into ECE cache from the persistence database and evicts some other data from ECE cache. This ensures that the maximum limit is not exceeded. Later, when you restart the ECE system, ECE loads the specified number of most recently used data into the ECE cache. In case of a node failure, ECE ensures high availability by distributing the data in the failed node to other nodes that are up and running.

For example, consider the following scenario:

- There are 1000 customers, and you want to load only 60 percent of your customers' data into ECE cache (that is, data from 600 customers).
- There are three ECE server machines and each machine has two charging server nodes up and running. In total, there are 6 charging server nodes that are available to load data.

In this case, you can set the **HighUnits** to 150 and **LowUnits** to 100. When the charging server nodes are started, each node loads data for only 100 customers into ECE cache, which is in total 600 customers data, the minimum limit. The remaining quota (which is 50 per node) is reserved for the data migrated from other nodes in case of machine or node failures. If one of the machines fails, the data from the two charging server nodes in that machine, which is for 200 customers, is distributed evenly to the nodes in the other two machines. This increases the load of each node to 150 units, which is the maximum limit for loading data into ECE cache. In case of ECE restart, each node loads only 100 of the most recently used customer data into ECE cache.

To enable partial loading of data:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **BRMFederatedCache**.
4. Expand the cache for which you want to enable partial loading such as Customer.
5. Expand **back**.
6. Expand **Attributes**.
7. Specify the following values:
   - **HighUnits**: Specify the maximum units (for each node) of data that can be stored in ECE cache.

     To enable partial loading, specify a positive numerical value. Setting this zero (**0**) specifies to store all data in the cache.
   - **LowUnits**: Specify the minimum number of records (for each node) to be loaded into the cache at the time of initial loading or ECE restart. This value must be less than the **HighUnits** value.

     If this entry is set to zero (**0**) and **HighUnits** is set to a non-zero value, 80 percent of the **HighUnits** value is considered for reloading the data.

Querying Persistence Database

To query the persistence database, run the following command:

```
./query.sh -p
```

For example, you can use this utility to verify if the data is loaded from persistence or the original source.

See "query" in *BRM Implementing Charging* for more information.

70

Managing Failed Customer Data Updates

Learn how to configure and manage failed customer data updates in Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

Managing Failed Customer Data Updates

The following are examples of when customer data updates from BRM might fail:

- A delay in receiving PDC pricing data

  A PDC delay where ECE has not received a charge offer or discount offer from PDC but did receive the offer external ID from BRM. Since the offer is not loaded into ECE because of the PDC delay, the external ID update does not find the offer in the cache. Consequently, the offer update fails.

- Configuration errors in customer data
- Errors in management requests associated with the rerating process

  Management-type requests for the rerating process, such as PrepareToRerate and RerateCompleted, may fail.

Customer data updates are placed in a suspense queue. As part of the post-installation tasks in an ECE installation, you create the suspense queue for BRM Gateway to collect the failed customer data updates and you create a queue table called **ifw_sync_sus**.

Once you have collected the failed updates, you view them in the BRM Gateway suspense queue and manually reprocess. When failed updates are moved to the suspense queue, error messages are displayed in the **customerUpdater.log** files in the *ECE_home***logs** directory.

To view the failed updates in the suspense queue:

1. Map the suspense queue to the **ifw_sync_sus** table.
2. Run the **select user_data from ifw_sync_sus** query to view the failed events after connecting to the SQL database.

Once you view the events in the suspense queue and you identify the cause of failure, you propagate the events from the suspense queue to the BRM Oracle DM database queue with the **events_propagate_utility** script. The Customer Updater reprocesses the events. For more information, see "events_propagate_utility.pl".

Moving Failed Data Updates From the Suspense Queue

ECE automatically moves the failed updates from the suspense queue at regular intervals to BRM Gateway. You can change the interval at which the requests are moved from the suspense queue and the maximum number of requests that can be moved at a time.

Typically, ECE moves the update failed due to network failure or other exceptions in BRM; for example, credit limit breach. However, if the update failed due to incorrect message format, you must manually fix the format and then move the failed update. You can perform this by using WebLogic Server Administration Console.

Before fixing the message format, you can increase the interval for moving the failed updates to BRM Gateway, as required, so that the requests are moved to BRM Gateway only after you fix the message format. See "Modifying the Interval and Batch Size for Moving Failed Data Updates".

To fix the message format and move the failed update manually, do the following in WebLogic Server Administration Console:

1. Export the update request into an XML file.

   For instructions, see the discussion about exporting messages in the WebLogic Server Administration Console Help.

2. Fix the message format by editing the XML file.
3. Import the fixed update request in the XML format.

   For instructions, see the discussion about importing messages in the WebLogic Server Administration Console Help.

4. Move the failed updates from the suspense queue by doing the following:
   a. Log on to the WebLogic server on which the BRM Gateway suspense queue resides.
   b. Log in to WebLogic Server Administration Console.
   c. In the **Services** section, select **JMS Modules.**
   d. In the **JMS Modules**, click **ECE Module**.
   e. In **Summary of Resources**, click **Suspense Queue**.
   f. Click **Monitoring**, and select **ECE!Suspense Queue**.
   g. Click **Show Messages**.

      The Summary of **JMS Messages** appears.

   h. In the **JMS Messages** table, select a message.
   i. Click **Move**.
   j. In the **NotificationServer** field, select **JMS Server**.
   k. In the **DestinationServer** field, select **Suspense Queue.**
   l. Click **Finish**.

      The failed update is sent back to the BRM Gateway suspense queue for reprocessing.

Modifying the Interval and Batch Size for Moving Failed Data Updates

To modify the interval and batch size for moving the failed data updates from the suspense queue to BRM Gateway:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.brmGatewayConfiguration**.
4. Expand **Attributes**.
5. Specify the values for the following attributes:
   - **brmSuspenseQueuePeriod.** Specify the interval (in milliseconds) for moving the failed update requests from the BRM Gateway suspense queue to BRM Gateway.
   - **jmsBatchSize.** Specify the maximum number of failed update requests to be moved from BRM Gateway suspense queue to BRM Gateway in a batch.

Purging Failed Data Updates From the Suspense Queue

To manually delete a failed update:

1. Log on to the WebLogic server on which the suspense queue resides.
2. Log in to WebLogic Server Administration Console.
3. In the **Services** section, select **JMS Modules.**
4. In the **JMS Modules**, click **ECE Module**.
5. In **Summary of Resources**, click **Suspense Queue**.
6. Click **Monitoring**, and select **ECE!Suspense Queue**.
7. Click **Show Messages**.

   The Summary of **JMS Messages** appears.
8. In the **JMS Messages** table, select a message.
9. click **Delete**.

The failed update is deleted from the suspense queue.

Loading Events Into the Database

After you view the events in the suspense queue and identify the cause of failure, you propagate the events from the suspense queue to the Account Synchronization Data Manager (DM) database queue with the **events_propagate_utility** script. Customer Updater reprocesses the events. You can also move events or purge data from the suspense queue by using the **events_propagate_utility** script. For more information, see "events_propagate_utility.pl".

71

Removing Data from the ECE System

Learn how to remove data from Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

Deleting Expired Data

ECE deletes expired historic data that is no longer used for processing customer updates and usage information. This includes expired objects for audit data, purchased charge offers, and so on. The clean-up occurs during the update and usage request processes. You can configure the retention time of expired objects in update requests but not in usage requests.

To configure the retention time (in days) of an expired object in an update request:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.expirationConfiguration**.
4. Expand **Attributes**.
5. Specify values for the expiration configuration attributes listed in Table 71-1.

   The maximum allowed retention time is 180 days.

Table 71-1 MBean Attributes to Configure Expired Object Retention Time

MBean Attribute	Default Retention Time (in Days)
expiredAuditRetentionIntervalInDays	60
expiredPurchasedProductRetentionIntervalInDays	30
expiredPurchasedAlterationRetentionIntervalInDays	30
expiredRatingProfileRetentionIntervalInDays	30
defaultExpirationRetentionIntervalInDays	30
defaultExpiredBalanceItemRetentionIntervalInDays	30

Expired audit objects share a single common retention time; other expired objects have individual retention times.

Expired balance items are configured at the balance element level. If there is no configuration for a given balance element, **defaultExpiredBalanceItemRetentionIntervalInDays** is used. Table 71-2 lists the expired balance elements and example retention times.

Table 71-2 Expired Balance Element Retention Times

Expired Balance Element	Retention Time (in Days)
FREE_MIN	60
BONUS_POINTS	15

Setting Eviction Policies for Cache

The identity cache stores customer public user identity information. The pricing cache stores pricing information, and the configuration cache stores configuration information. You can set eviction policies for these caches to remove entries from them when a maximum number of entries is reached. The default settings are:

- Eviction policy = **HYBRID**
- High-units = **20,500,000**

By default, the entries in the pricing-and-configuration near caches that have not been used for more than 10 days are evicted. You can reset the size limit for the front cache by changing the **front-size-limit** parameter for the pricing or configuration caches in the *ECE_home***/config/charging-cache-config.xml** file. Ensure that this parameter is set to a non-zero value.

For more information, see the Oracle Coherence documentation.

Purging Rated Events from the Oracle NoSQL Database

Rated Event Formatter purges rated events from the NoSQL database immediately after sending them to Rated Event Loader. You can retain events for a period of time before purging them. For example, you can retain them for an hour if legal requirements mandate that, or to give another program time to get data from the rated events.

To configure how rated events are purged from Oracle NoSQL database:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.ratedEventFormatters.***Instance_Name*, where *Instance_Name* is the name of the instance you want to configure.
4. Expand **Attributes**.
5. Set the **retainDuration** attribute to the number of seconds to wait before purging rated events.
6. Use Elastic Charging Controller to stop and restart the Rated Event Formatter instances that you configured.

The Rated Event Formatter instance purges the rated events and writes log information to the Rated Event Formatter log file.

About Purging Accounts from the ECE Cache

To optimize and efficiently manage memory, ECE allows you to purge inactive and dormant accounts from the ECE cache. You can purge the accounts synchronously from both BRM and ECE in real time using EM Gateway.

When you delete an account from the BRM database using the PCM_OP_CUST_DELETE_ACCT opcode, BRM sends the update request to ECE through EM Gateway. ECE processes the update request and purges the account from the ECE cache if there are no active sessions associated with that account. The BRM database and the ECE cache are updated within the same transaction in real time, ensuring that the BRM and ECE data remain synchronized.

When purging the account from the ECE cache, ECE purges the customer data associated with the account, which includes:

- Customer information
- Customer balance history
- Recurring bundle history
- Account top-up history
- Sharing customer state history
- Billing trigger cycle information
- Public user identity (PUID), if it is not associated with any other active account

To purge the accounts synchronously in BRM and ECE, you must enable synchronous updates in BRM and ECE. For more information, see "Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates" in *ECE Implementing Charging*.

Setting Retention Policies for ECE Tables with SecureFiles LOBs

By default, ECE stores data in the following tables as SecureFiles LOBs:

- BALANCE
- CUSTOMER
- CUSTOMERGROUP
- RATEDEVENT
- RECURRINGBUNDLEIDHISTORY
- TOPUPHISTORY

The data in these tables grow continuously and cannot be purged. To prevent these tables from growing too large, configure ECE to reuse space in these tables by setting their data retention policy to **NONE**.

To set retention policies for tables with SecureFiles LOBs:

1. Connect to the BRM database with SQL*Plus:

```
sqlplus userName@databaseAlias
Enter password: password
```

where:

- *userName* and *password* are the login name and password for connecting to the BRM database.
- databaseAlias is the BRM database alias.

2. Set the retention policy to **NONE**:

```
SQL> alter table tableName modify lob (c_lob) (retention NONE);
Table altered.
```

where *tableName* is the name of one of the tables that store data in SecureFiles LOBs.

For example, you would enter this for the RECURRINGBUNDLEIDHISTORY table:

```
SQL> alter table RECURRINGBUNDLEIDHISTORY modify lob (c_lob) (retention
NONE);
Table altered.
```

3. Type **exit** to quit SQL*Plus.

For more information about setting the retention of SecureFiles LOBs, see the "How to change retention of securefile Lob segment (Doc ID 2175438.1)" knowledge article on My Oracle Support.

72

Configuring ECE for a Multischema BRM Environment

Learn how to configure Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) in a multischema environment.

Topics in this document:

See also "Running Utilities in Multischema Systems" and "Managing a Multischema System".

Configuring ECE for a Multischema BRM Environment

To configure ECE for a multischema BRM environment:

- Configure a Rated Event Formatter instance to process rated events belonging to each BRM schema. See "Configuring Rated Event Formatter" in *ECE Implementing Charging*.
- Configure a Customer Updater instance for each schema. To configure multiple Customer Updater instances, do the following:
  - Use the ECC **AddNode** command to add a Customer Updater process for each BRM schema. For example, to configure three Customer Updater processes, add three nodes named **customerLoader**, **customerLoaderSchema2**, and **customerLoaderSchema3**. See "Adding Nodes".
  - Use a JMX editor to configure the Customer Updater instance MBeans. Use a different value in the **schemaNumber** attribute for each Customer Updater process. See "Configuring Customer Updater".

> **Note:**
>
> To support account migration, the configuration for each Customer Updater instance must specify the name of the Account Migration Manager (AMM) acknowledgment queue in the **amtAckQueueName** attribute. (All BRM database schema use the same AMM acknowledgment queue).
>
> For more information, see *BRM Moving Accounts between Database Schemas.*

73

Backing Up and Restoring ECE

Learn about the tasks that you perform to back up and restore Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

About Backing Up and Restoring ECE

After installing or upgrading an ECE system, perform one of the following backups as a safety measure:

- Standard configuration backup. See "About Standard Configuration Backup".
- Complete system backup. See "About Complete System Backup".

Repeat the backup process whenever you make any changes in the data or configuration files. If you do not back up the ECE system regularly, you need to reinstall and reconfigure ECE if the system is corrupted due to operational or system errors. Reinstalling and reconfiguring eliminates any chance of recovering and reprocessing data processed by the ECE system at the time of the error.

About Standard Configuration Backup

The standard configuration backup is a backup of the ECE configuration files and ECE installation area (the ECE installation directory and its content: *ECE_home*). You can perform this backup as soon as you install and configure ECE. In particular, make sure you back up all customized files. For example, you need to perform this procedure if you customized any settings in the *ECE_home***/config/management/charging-settings.xml** file.

Repeat the backup process whenever you customize or update ECE configuration files. For instructions on backing up the standard configuration, see "Backing Up Standard Configuration".

When you complete a standard configuration backup, you can use the backup configuration directory and installation area to restore your system when needed. For instructions on restoring the standard configuration, see "Restoring a Standard System Configuration".

> **Note:**
>
> Store this backup in a safe location. The data in these files are necessary if you encounter any operational or system issues.

If you require ECE technical support, email a copy of your backup configuration directory to your Oracle Global Support representative. If you want to send a copy of your configuration directory, use the tar command to create an archived version of that directory.

About Complete System Backup

The complete system backup is a backup of the complete ECE system. You can perform the complete system backup by creating a complete offline backup of the following:

- ECE configuration files and ECE installation area. See "Backing Up Standard Configuration".
- Oracle NoSQL database configuration and Oracle NoSQL database installation area. See "About Backing Up Oracle NoSQL Installation and Configuration".

> **Note:**
>
> If you have multiple ECE charging server nodes configured in your system, perform the backup on the driver machine. Make sure that you repeat the backup process regularly.
>
> Store this backup in a safe location. The data in these files are necessary if you encounter any operational or system issues.

Typically, ECE files generated at run time (such as Rated Event (RE) Loader files (rated event data), RE loader control files, log files, and Coherence-related files) are automatically recovered when you restore your ECE system. For example, if the RE loader control files are not available in your restored ECE system, Rated Event Formatter automatically generates the RE loader control files based on the event specification data loaded into ECE from PDC. Therefore, you need not create an offline backup of the ECE files generated at run time.

However, if you want to avoid the risk of losing any data, you can create a backup of RE loader files. See the discussion about backing up RE Loader files in *BRM Configuring Pipeline Rating and Discounting*.

For instructions on restoring your complete ECE system backup, see "Restoring a Complete System Backup".

About Backing Up Oracle NoSQL Installation and Configuration

When data persistence is disabled, ECE uses Oracle NoSQL Database to temporarily store rated event information. Specifically, ECE uses the multiple-node Oracle NoSQL data store configuration in production environment for storing the ECE data. Make

sure you create a backup of Oracle NoSQL installation area (the Oracle NoSQL installation directory and its content) and the multiple-node Oracle NoSQL data store configuration. You can perform this backup as soon as you install and configure the Oracle NoSQL database. Repeat the backup process whenever you customize or update the Oracle NoSQL data store configuration.

For instructions on backing up the Oracle NoSQL Database installation, see "Backing Up the Oracle NoSQL Database Installation".

You can also make a complete offline backup of Oracle NoSQL Database using the appropriate backup tools for your database version and ensure that the backup is completely valid and usable. The backup must contain both the database definition and all the database contents. See the Oracle NoSQL Database documentation for more information on performing full database backups.

You can make a complete offline backup of the multiple-node Oracle NoSQL data store configuration by taking snapshots of the nodes. See the discussion about backing up the Oracle NoSQL data store in the Oracle NoSQL database documentation for instructions on backing up an Oracle NoSQL Database data store.

Backing Up Standard Configuration

To back up standard configuration:

1. On the driver machine, go to the *ECE_home* directory.
2. Copy the content of the *ECE_home* directory to a new directory:

   ```
   cp -R ECE_home NewName
   ```

   where *NewName* is the name for the new directory.

   > **Note:**
   >
   > You can remove the ECE log files and Coherence-related files from the backup directory.

3. Create an archive of the entire directory:

   ```
   tar cvf NewName.tar.gz NewName
   ```

4. Store the backup copy in a location outside of the ECE system:

   ```
   mv NewName.tar.gz New_Directory
   ```

   where *New_Directory* is the new location that is outside of the ECE system.

   A compressed TAR file, of all copied files, is created with the extension **tar** in the new location specified (for example, ece_backup/ece_home.tar.gz).

Backing Up the Oracle NoSQL Database Installation

To back up a Oracle NoSQL database installation:

1. On the machine in which the Oracle NoSQL database is installed, go to the *Oracle_NoSQL_DB_Home* directory.

2. Copy the content of the *Oracle_NoSQL_DB_Home* directory to a new directory:

   ```
   cp -R Oracle_NoSQL_DB_Home NewName
   ```

   where *NewName* is the name for the new directory.

3. Create an archive of the entire directory:

   ```
   tar cvf NewName.tar.gz NewName
   ```

4. Store the backup copy in a location outside of the ECE system:

   ```
   mv NewName.tar.gz New_Directory
   ```

   where *New_Directory* is the new location that is outside of the ECE system.

   A compressed TAR file, of all copied files, is created with the extension **tar** in the new location specified (for example, Oracle_noSQL_db_backup/ noSQL_db_bkp.tar.gz).

Restoring a Standard System Configuration

To restore an ECE system backup:

1. In PDC, publish all the PDC pricing data (the metadata, setup, pricing, and profile data) from the PDC database to ECE by running the following command:

   ```
   ImportExportPricing -publish -metadata -config -pricing -profile -target
   [ece]
   ```

2. Delete or rename the damaged *ECE_home* directory:

   - To delete, run the following command:

     ```
     rm -R ECE_home
     ```

   - To rename, run the following command:

     ```
     mv ECE_home New_Name
     ```

3. Retrieve the backup tar file of the *ECE_home* directory.

4. Extract from the tar file the backup copy of the directory:

   ```
   tar xvf ECE_home.tar.gz
   ```

   The command recreates the copy of the *ECE_home* directory.

5. On the driver machine, go to the *ECE_home***/bin** directory.

6. Start ECC:

   ```
   ./ecc
   ```

7. Enable real-time synchronization of BRM and ECE customer data updates. See the discussion about configuring ECE for synchronizing BRM and ECE customer data in real time in *BRM ECE Implementing Charging* for more information.

8. Start ECE processes and gateways in the following order:

> **Note:**
>
> Depending on your installation, you can start Diameter Gateway, RADIUS Gateway, both Diameter Gateway and RADIUS Gateway, or none.

```
start server
start configLoader
start pricingUpdater
start customerUpdater
start emGateway
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
```

All data is now back in the ECE data grid.

Real-time-data updates, which had been temporarily disrupted due to the shutdown, are processed upon restart.

Restoring a Complete System Backup

To restore a complete ECE system backup:

1. In PDC, publish all the PDC pricing data (the metadata, setup, pricing, and profile data) from the PDC database to ECE by running the following command:

   ```
   ImportExportPricing -publish -metadata -config -pricing -profile -target [ece]
   ```

2. Delete or rename all the damaged directories; for example, *ECE_home* and *Oracle_NoSQL_DB_Home*:

   - To delete, run the following command:

     rm -R *Directory_Name*

   - To rename, run the following command:

     mv *Directory_Name New_Name*

3. Retrieve all the backup tar files. For example, ece_home.tar.gz and noSQL_db_bkp.tar.gz.

4. Extract from the tar files the backup copy of the directories:

   tar xvf *Directory_Name.tar.gz*

   The command recreates the directories in your restored installation directory.

5. Repeat steps 2 to 4 to restore all the damaged directories.

6. On the driver machine, go to the *ECE_home***/bin** directory.

7. Start ECC:

   ```
   ./ecc
   ```

8. Enable real-time synchronization of BRM and ECE customer data updates. See the discussion about configuring ECE for synchronizing BRM and ECE customer data in real time in *BRM ECE Implementing Charging* for more information.

9. Start ECE processes and gateways in the following order:

> **Note:**
>
> Depending on your installation, you can start Diameter Gateway, RADIUS Gateway, both Diameter Gateway and RADIUS Gateway, or none.

```
start server
start configLoader
start pricingUpdater
start customerUpdater
start emGateway
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
```

All data is now back in the ECE data grid.

Real-time-data updates, which had been temporarily disrupted due to the shutdown, are processed upon restart.

74

Configuring ECE for High Availability

Learn how to configure Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) for high availability.

Topics in this document:

Configuring Backup Driver Machine for High Availability

You can configure a backup driver machine that takes over when the primary machine goes down or fails. You can configure the backup driver machine by setting the **backupDriverIP** entry in the *ECE_home*/**config/ece.properties** file. When the backup driver machine is configured:

- The ECE configuration in both the driver machine and the backup driver machine is updated each time an ECE MBean is set.
- If the primary or backup driver machine goes down or if it is not reachable, the ECE configuration in the driver machines is not updated when an ECE MBean is set.

If you have a driver machine and multiple server machines in your system, you can configure one of the existing server machines to be the new driver machine or the backup driver machine. Before you configure the driver machine:

- Ensure that the selected machine has a JMX-management-enabled server node running.
- Ensure that the two-way password-less SSH connection is established between the selected machine and all the other machines in your topology.
- Ensure that the primary driver machine is added to the ECE topology. This ensures that the primary driver machine is also considered for synchronization when the **sync** command is run.

1. On the selected machine, do one of the following:
   - To configure the selected machine as the new driver machine, run the following command:

     setDriverMachine -name driverIP -ip *Driver_IP_Address*

where *Driver_IP_Address* is the IP address of the machine selected to be the new driver machine.

Running this command without the **-name driverIP** parameter configures the selected machine as the primary driver machine.

- To configure the selected machine as the backup driver machine, run the following command:

```
setDriverMachine -name backupDriverIP -ip IP_Address
```

where *Backup_IP_Address* is the IP address of the machine selected to be the backup driver machine.

The selected machine is assigned as the new or backup driver machine and a success message appears.

2. Verify that the new or backup driver machine is performing the relevant operations.

About Configuring Data Updaters for High Availability

You can configure and start multiple additional instances of the following components in the same system or different ECE systems (such as backup machines) to ensure high availability:

- Rated Event Formatter
- BRM Gateway
- EM Gateway
- Customer Updater
- Pricing Updater

All the instances of the component configured for high availability can run at the same time with one instance always in the active mode and other instances in the standby mode. When the active instance of the component goes down due to system failure, one of the instances in the standby mode automatically becomes active and the instance that failed becomes standby. All the other instances continue to remain in the standby mode. You can configure additional instances of these components by using ECE Mbeans and ECE topology settings.

You can monitor the status of each component and server configured for high availability by using ECE Mbeans. For more information, see "Viewing ECE High Availability Status".

Configuring Customer Updater for High Availability

To configure a Customer Updater instance for high availability:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations**.
4. Expand **Operations**.
5. Click **addoracleQueueConnectionConfiguration**.

6. Set the **name** attribute to the name of the additional Customer Updater instance; for example, CustomerUpdater2.
7. Open the *ECE_home*/**config/eceTopology.conf** file.
8. Add a row to the file for the Customer Updater instance.

   Use a unique name for each instance so that Elastic Charging Controller (ECC) can distinguish between them. (All Customer Updater instances use the same role).

   For example, to configure three Customer Updater instances, add three rows in which the **name** value is **customerUpdater**, **customerUpdater2**, and **customerUpdater3** respectively, and the **role** value for all three rows is **customerUpdater**.
9. Save and close the file.
10. Repeat step 1.
11. Expand **charging.connectionConfigurations.customerUpdater***n*, where *n* is the number that represents the additional instance; for example, CustomerUpdater2.
12. Expand **Attributes**.
13. Specify the attributes for the additional Customer Updater instance.

> **Note:**
>
> You can configure multiple Customer Updater instances to connect to the same BRM schema.

For information on the attributes, see "Configuring Customer Updater".

Configuring Pricing Updater for High Availability

To configure a Pricing Updater instance for high availability:

1. Open the *ECE_home*/**config/eceTopology.conf** file.
2. Add a row to the file for the Pricing Updater instance.

   Use a unique name for each instance so that Elastic Charging Controller (ECC) can distinguish between them. (All Pricing Updater instances use the same role).

   For example, to configure three Pricing Updater instances, add three rows in which the **name** value is **pricingUpdater**, **pricingUpdater2**, and **pricingUpdater3** respectively, and the **role** value for all three rows is **pricingUpdater**.
3. Save and close the file.

Configuring BRM Gateway for High Availability

To configure a BRM Gateway instance for high availability:

1. Open the *ECE_home*/**config/eceTopology.conf** file.
2. Add a row to the file for the BRM Gateway instance.

   Use a unique name for each instance so that Elastic Charging Controller (ECC) can distinguish between them. (All BRM Gateway instances use the same role).

For example, to configure three BRM Gateway instances, add three rows in which the **name** value is **brmGateway**, **brmGateway2**, and **brmGateway3** respectively, and the **role** value for all three rows is **brmGateway**.

3. Save and close the file.
4. Specify the attributes for the additional BRM Gateway instance by using ECE Mbeans.

   For instructions, see "Configuring Multiple BRM Gateway Instances" in *ECE Implementing Charging*.

Configuring EM Gateway for High Availability

To configure External Manager (EM) Gateway for high availability:

1. Open the *BRM_home***/sys/cm/pin.conf** file in a text editor.
2. Add additional EM Gateway pointers to the file to use for failover. For example:

```
-cm ece_real_time_sync_db_no 0.0.9.8
-cm em_group ece PCM_OP_ECE_PUBLISH_EVENT
-cm em_pointer ece ip emGateway_host1 emGateway_port1
-cm em_pointer ece ip emGateway_host1 emGateway_port2
-cm em_pointer ece ip emGateway_host2 emGateway_port3
-cm em_pointer ece ip emGateway_host2 emGateway_port4
```

   where

   - *emGateway_host1* is the name or IP address of the server on which one instance of EM Gateway is running.
   - *emGateway_port1* is the primary port number for EM Gateway on host 1.
   - *emGateway_port2* is a backup port number for EM Gateway on host 1.
   - *emGateway_host2* is the name or IP address of the server on which another instance of EM Gateway is running. (EM Gateway can run on multiple hosts.)
   - *emGateway_port3* is the primary port number for EM Gateway on host 2.
   - *emGateway_port4* is a backup port number for EM Gateway on host 2.
3. Set the value of the following entry to 2 or higher:

```
-cm cm_op_max_retries 2
```

   This entry specifies the maximum number of times an opcode is retried in the CM. The default value is **1**. For high availability, the value must be at least **2**.
4. Save and close the file.
5. Restart the CM.

Configuring Rated Event Formatter for High Availability

To configure a Rated Event Formatter instance for high availability:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.

3. Expand **charging.ratedEventFormatters.***Instance_Name*, where *Instance_Name* is the name of the instance you want to configure.
4. Expand **Attributes**.
5. Specify values for **name** and **partition** and for any remaining attributes you need to set for the instance. See
6. Change directory to the *ECE_home***/bin** directory.
7. Start ECC:

   ```
   ./ecc
   ```

8. Start Rated Event Formatter instances that you configured. See "Starting ECE".

Viewing ECE High Availability Status

To view the ECE high availability status:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Monitoring** node.
3. Expand **HighAvailability**.
4. Expand **Attributes**.

   The **ActiveInstances** attribute displays the active instance of each component configured for high availability.

   The **IsActive** attribute displays **true** or **false** to indicate whether the connected instance is an active instance or a passive instance. The **IsActive** value is displayed only for the components or sevrers configured for high availability.

   The **PassiveInstances** attribute displays all the passive instances of each component configured for high availability.

Configuring Additional Data Storage Node Connections for High Availability

If you are using Oracle NoSQL Database for storing rated events, you can configure additional data storage node connections to allow failover. When ECE processes are restarted, if one of the configured data storage nodes goes down, ECE processes can connect to any other configured data storage node that is currently running.

To configure additional data storage node connections:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.connectionConfigurations.***noSQLConnectioninstance*, where *noSQLConnectioninstance* is the name of the Oracle NoSQL database instance; for example, **noSQLConnection1**.
4. Expand **Attributes**.

5. Specify the value for the **dataStoreConnection** attribute. Enter the *hostname*:*port* for connecting to the Oracle NoSQL database instance. The default is "localhost:5000" for connecting to a standalone Oracle NoSQL database instance. You can add additional data store connections separated by a comma. For example:

```
server1:5000,server2:6000,server3:7000
```

75

Configuring ECE for Disaster Recovery

Learn how to configure Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) for disaster recovery.

Topics in this document:

About Disaster Recovery

Disaster recovery provides continuity in service for your customers and guards against data loss if a system fails. In ECE, you implement disaster recovery by configuring multiple load-sharing production sites and backup sites. If a production site fails, other load-sharing production sites or a backup site takes over. The production and backup sites are at geographically separate locations.

ECE supports the following types of disaster recovery configurations:

- Active-cold standby
- Active-warm standby
- Active-hot standby
- Segmented active-active
- Active-active

> **Note:**
>
> The active-warm standby configuration is supported only when data persistence is enabled in ECE. The remaining disaster recovery configurations can be used when data persistence is enabled or disabled.

In the disaster recovery deployment modes described in the following sections, BRM, PDC and OCOMC are always deployed in active-standby mode. Only ECE can be deployed in an active-standby mode or in an active-active mode.

Table 75-1 lists the different disaster recovery deployment modes for BRM, PDC, OCOMC, and ECE.

Table 75-1 Deployment modes for BRM, PDC, OCOMC, and ECE

Deployment Mode	BRM	PDC	OCOMC	ECE
Active-standby	Yes	Yes	Yes	Yes
Active-active	No	No	No	Yes

About the Active-Cold Standby System

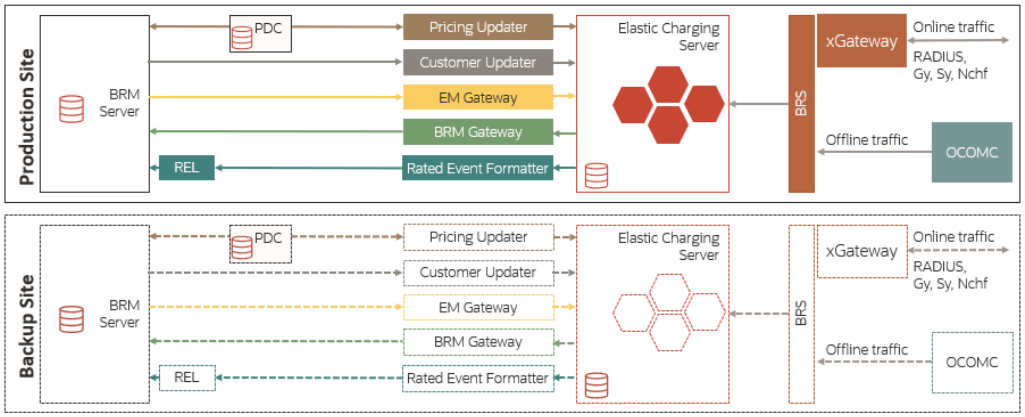
The active-cold standby configuration comprises an active production site and one or more idle backup sites. This system requires starting the backup site manually when the production site fails, which might cause a delay in bringing up the backup site to full operational capability. Use this type of system when some downtime is acceptable.

In the active-cold standby configuration, only the production system runs. The backup system is not started, and no data is replicated while the production system operates.

To switch to the backup system, you must start it and configure it to process usage requests.

Figure 75-1 shows the architecture for an active-cold standby disaster recovery system.

Figure 75-1 Architecture of an Active-Cold Standby Disaster Recovery System



A few critical aspects of the architecture are:

- Production sites and backup sites include the BRM, ECE, and PDC databases. However, for each type, you manage the configuration differently.
- In an active-cold standby configuration, only the production system runs. The backup system does not start, and the data is not replicated when the production system operates.
- To switch to the backup system, you must start it and configure it to process usage requests.

If you are using the Oracle NoSQL database for storing rated events, the ECE data in the Oracle NoSQL database data store is replicated from the production site to the backup site by using the primary and secondary Oracle NoSQL database data store nodes. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".

Configuring an Active-Cold Standby System

To configure an active-cold standby system:

1. In the production site and backup sites, do the following:
   - Install ECE and other components required for the ECE system. For more information, see "Installing Elastic Charging Engine" in *ECE Installation Guide*.

     > **Note:**
     >
     > If persistence is enabled in ECE, ensure that the name of the cluster and ECE components are identical across sites.

   - Ensure that any customizations to configuration files and extension implementation files in your production site are reapplied to the corresponding default files on the backup sites.
2. In the production site, start ECE. See "Starting ECE" for more information.

Failing Over to a Backup Site

You enable a backup site to take over the role of the production site. You can then use the backup site as the production site or switch back to the restored production site.

To fail over to a backup site:

1. If persistence is enabled in ECE, switch the ECE persistence database in the backup or remote production sites in the production system based on your Oracle Active Data Guard configuration. For more information on performing a failover, see "Failovers" in *Oracle Data Guard Concepts and Administration*.
2. On the backup site, start ECE. See "Starting ECE" for more information.
3. Verify that ECE is connected to BRM and PDC on the backup site by performing the following:

   > **Note:**
   >
   > If only ECE in the production site failed and BRM and PDC in the production site are still running, you must change the BRM, PDC, and Customer Updater connection details on the backup site to connect to BRM and PDC in the production site.

   - Verify that the details about JMS queues for Pricing Updater on the backup site are specified in the ECE configuration MBeans.
   - Use a JMX editor to verify that the BRM connection details on the backup site are provided in the ECE configuration MBeans.

4. Load the pricing and customer data into ECE.

> **Note:**
>
> If ECE persistence is enabled, after the ECE persistence database failover, the pricing data is automatically loaded in the ECE cache from the ECE persistence database. The customer data is loaded in the ECE cache on demand from the ECE persistence database during usage processing.

5. Ensure that the network clients route all requests to the backup site.

> **Note:**
>
> Information such as the balance, configuration, and rated event data still in the ECE cache when the production site failed is lost.

The former backup site is now the new production site.

> **Note:**
>
> If you switch back to the original production site in an active-cold standby system, any data in memory is lost. Oracle does not recommend switching back to the original production site in an active-cold standby system.

About the Active-Warm Standby System

The active-warm standby configuration consists of an active production site and one or more active backup sites. ECE cache, BRM, and PDC data are asynchronously replicated from the production site to the backup sites using Oracle Active Data Guard. The ECE configuration and mediation specification data are not replicated between participant sites. All ECE processes on the backup site are started in standby mode. When the production site fails, ECE requests are diverted from the production site to the backup sites. This type of system requires a minimal amount of downtime.

In the active-warm configuration, the backup system is started and running, but no usage requests are processed by it. The Pricing Updater, Customer Updater, EM Gateway, and BRM Gateway (if a BRM instance is available on the backup site) are configured and started as standby instances in the usage processing state, but these instances are not configured to load data. In addition, the Rated Event Formatter is configured, but not started on the backup system. All changes to customer and pricing data are made on the production system and replicated to the ECE persistence database on the backup system.

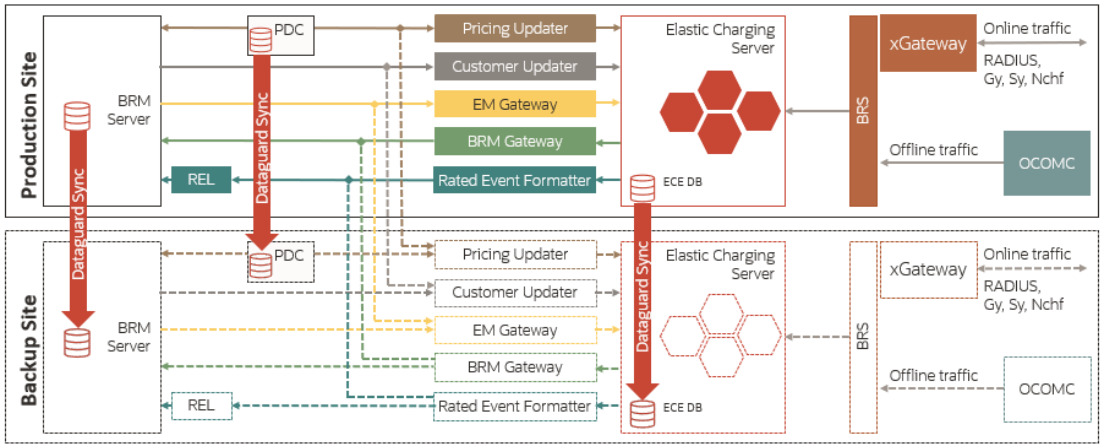
When you switch to the backup system, usage request processing is automatically enabled and the standby instances become the primary instances. These processes are connected to the BRM and PDC systems on the backup system.

On the backup system, the pricing data is loaded in the ECE cache from the ECE persistence database, and the customer data is loaded on demand during usage processing. However, you must start the Rated Event Formatter and ensure that it is connected to the BRM system on the backup system.

You use the Secure Shell (SSH) File Transfer Protocol (SFTP) utility to replicate CDR files generated by Rated Event Formatter between participant sites.

Figure 75-2 shows the architecture for an active-warm standby disaster recovery system.

Figure 75-2 Architecture of an Active-Warm Standby Disaster Recovery System



A few key aspects of the architecture are:

- Production sites and backup sites include the BRM, ECE, and PDC databases. However, for each type, you manage the configuration differently.
- To switch to the backup system, you must start the backup system and configure it to process usage requests.
- If only ECE cluster fails in the production site, the ECE cluster in the backup site is used with BRM from the production site. If ECE and BRM fail in the production site, you must switch over to the backup site.

Configuring an Active-Warm Standby System

> **Note:**
>
> In an active-warm standby system, ensure that the name of the cluster and ECE components are identical across sites.

To configure an active-warm standby system:

1. In the primary production site, do the following:
   a. Configure ECE to persist the cache and rated event data in the ECE persistence database. See "Managing Persisted Data in the Oracle Database" for more information.

b. Configure the ECE components (such as Customer Updater, Pricing Updater, EM Gateway, and so on).

c. Start ECE. See "Starting ECE" for more information.

2. On the backup site, do the following:

a. Configure ECE to persist the cache and rated event data in the ECE persistence database. See "Managing Persisted Data in the Oracle Database" for more information.

b. Configure the ECE components.

c. Start ECC:

```
./ecc
```

d. Start the charging server nodes:

```
start server
```

e. Start the ECE components in standby mode.

Failing Over to a Backup Site

You enable a backup or remote production site to take over the role of the primary production site. After the original production site is fixed, you can return the sites to their original state by switching the workload back to the original production site.

To fail over to a backup site:

1. Enable the ECE persistence database in the backup or remote production sites in the primary role. For more information on performing a failover, see "Failovers" in *Oracle Data Guard Concepts and Administration*.
2. Start BRM and PDC.
3. Ensure that the replicated data in the ECE persistence database is loaded into the ECE cache and that the ECE processes (Customer Updater, Pricing Updater, and so on) are changed to the primary instances.

   All ECE and pricing data is now back in the ECE grid in the backup or remote production site.
4. Start Rated Event Formatter by running the following command:

   ```
   start ratedEventFormatter
   ```

   All rated events are now processed by Rated Event Formatter in the backup or remote production site.
5. Stop and restart BRM Gateway.
6. Ensure that the network clients route all requests to the backup or remote production site.

Switching Back to the Original Production Site

You restart the restored production site and restore the data in the ECE persistence database on the backup system by using Oracle Active Data Guard.

To switch back to the original production site:

1. Install ECE and other required components in the original primary production site. For more information, see "Installing Elastic Charging Engine" in *ECE Installation Guide*.

   > **Note:**
   >
   > If only ECE in the original primary production site failed and BRM and PDC in the original primary production site are still running, install only ECE and provide the connection details about BRM and PDC in the original primary production site during ECE installation.

2. Switch the ECE persistence database in the original production site in the primary role. For more information on performing a switchover, see "Switchovers" in *Oracle Data Guard Concepts and Administration*.
3. On the machine on which the Oracle WebLogic server is installed, verify that the JMS queues have been created for loading pricing data and for sending event notification, and that JMS credentials have been configured correctly.
4. Go to the *ECE_home***/bin** directory.
5. Start ECC:

   ```
   ./ecc
   ```

6. Start the charging server nodes:

   ```
   start server
   ```

7. Ensure that the ECE cache data is replicated to the original production site by using Oracle Active Data Guard.

   > **Note:**
   >
   > In the original primary production site, if only ECE has failed, but the ECE persistence database is still running, install only ECE and provide the connection details to the ECE persistence database in the original primary production site.

8. Ensure that the ECE processes on the original primary production site (Customer Updater, Pricing Updater, and so on) are changed to the primary instances. All ECE and pricing data is now back in the ECE grid in the original primary production site.
9. Start the following ECE processes and gateways:

   ```
   start brmGateway
   start diameterGateway
   start radiusGateway
   ```

10. Stop and restart Pricing Updater, Customer Updater, and EM Gateway in standby mode in the new primary production site and then start them in the original primary production site.
11. Stop the Rated Event Formatter in the new primary production site and then start it in the original primary production site.
12. Stop RE Loader in the new primary production site and then start it in the original primary production site.

13. Stop and restart BRM Gateway in both the new primary production site and the original primary production site.
14. Ensure that the network clients route all requests to the original primary production site.

    The roles of the sites are now reversed to the original roles.

About the Active-Hot Standby System

The active-hot standby configuration consists of an active production site and one or more active backup sites. ECE data is asynchronously replicated from the production site to the backup sites. When the production site fails, ECE requests are diverted from the production site to the backup sites. This type of system requires a minimal amount of downtime.

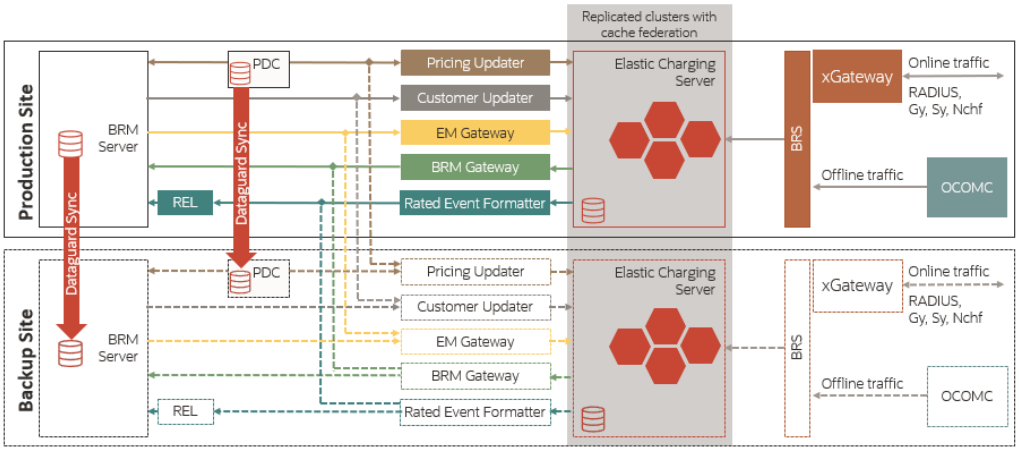
In the active-hot standby configuration, the backup system is started and running, but no usage requests are processed by it. Additionally, the Customer Updater, Pricing Updater, and EM Gateway are configured, but not started on the backup system. All changes to customer and pricing data are made on the production system, and replicated on the backup system.

To switch to the backup system, you enable usage request processing for the system and then start the Customer Updater, Pricing Updater, and EM Gateway. These processes are connected to the BRM and PDC systems on the backup system.

You use the Secure Shell (SSH) File Transfer Protocol (SFTP) utility to replicate CDR files generated by Rated Event Formatter between participant sites.

Figure 75-3 shows the architecture for an active-hot standby disaster recovery system.

Figure 75-3 Architecture of an Active-Hot Standby Disaster Recovery System



A few key aspects of the architecture are:

- The production site is active and the backup site is in standby mode. The backup site replicates data from the production site. Also, by starting processes on the backup site, the production site retrieves and processes the data when there is a failover.

- If only ECE cluster fails in the production site, the ECE cluster in the backup site is used with BRM from the production site. If ECE and BRM fail in the production site, you must switch over to the backup site.

If you are using the Oracle NoSQL database for storing rated events, the ECE data in the Oracle NoSQL database data store is replicated from the production site to the backup site by using the primary and secondary Oracle NoSQL database data store nodes. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".

Configuring an Active-Hot Standby System

When configuring active-hot standby with ECE persistence enabled, consider the following points:

- Do not set the **loadConfigSettings** parameter in the secondary site, as you need to load **appConfig** on both sites. For more information, see "Specifying Driver Machine Properties" in *ECE Installation Guide*.
- Synchronize the wallet correctly on both sites. Whenever passwords are updated, you need to copy the wallet manually to the other site.
- Clean the persistence store when you are restarting the system. Whenever a site fails to respond and if you want to bring up the site using the **gridsync** utility from the other site, you must:
  - Start the site from a clean state in the same way the secondary cluster started.
  - Clear old data from the persistence store.
- Run only one **RatedEventFormatter** instance among the sites for a given RatedEventFormatter configuration.

To configure an active-hot standby system:

1. In the primary production site, do the following:
   a. Configure primary and secondary Oracle NoSQL database data store nodes. For more information, see "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide*.
   b. Configure the ECE components (Customer Updater, EM Gateway, and so on).
   c. Add details about all participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home*/**config/charging-coherence-override-prod.xml**).

      To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home*/**config/ece.properties** file.

      Table 75-2 provides the federation configuration parameter descriptions and default values.

Table 75-2 Federation Configuration Parameters

Name	Description
name	The name of the participant site. **Note:** The name of the participant site must match the name of the cluster in the participant site.
address	The IP address of the participant site.

Table 75-2 (Cont.) Federation Configuration Parameters

Name	Description
port	The port number assigned to the Coherence cluster port of the participant site.
initial-action	Specifies whether the federation service should be started for replicating data to the participant sites. Valid values are: • **start**: Specifies that the federation service has to be started and the data must be automatically replicated to the participant sites. • **stop**: Specifies that the federation service has to be stopped and the data must not be automatically replicated to the participant sites. **Note:** Ensure that this parameter is set to **stop** for all participant sites except for the current site. For example, if you are adding the backup or remote production sites details in the primary production site, this parameter must be set to **stop** for all backup or remote production sites.

d. Start ECE. See "Starting ECE" for more information.

2. On the backup site, do the following:

   a. Configure primary and secondary Oracle NoSQL database data store nodes. For more information, see "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide*.

   b. Configure the ECE components (Customer Updater, EM Gateway, and so on).

   c. Set the following parameter in the *ECE_home***/config/ece.properties** file to **false**:

      ```
      loadConfigSettings = false
      ```

      The application-configuration data is not loaded into memory when you start the charging server nodes.

   d. Add details about all participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging-coherence-override-prod.xml**).

      To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file. Table 75-2 provides the federation configuration parameter descriptions and default values.

   e. Start the Elastic Charging Controller (ECC):

      ```
      ./ecc
      ```

   f. Start the charging server nodes:

      ```
      start server
      ```

3. On the primary production site, run the following commands:

   ```
   gridSync start
   gridSync replicate
   ```

   The federation service is started and all existing data is replicated to the backup or remote production sites.

4. On the backup sites, do the following:

a. Verify that the same number of entries as in the primary production site are available in the customer, balance, configuration, and pricing caches in the backup or remote production sites by using the **query.sh** utility.

b. Verify that the charging server nodes in the backup or remote production sites are in the same state as the charging server nodes in the primary production site.

c. Configure the following ECE components by using a JMX editor:

- Rated Event Formatter
- Rated Event Publisher
- Diameter Gateway
- RADIUS Gateway
- HTTP Gateway

The federation service is started to replicate the data from the backup or remote production sites to the primary production site.

After starting Rated Event Formatter in the remote production sites, ensure that you copy the CDR files generated by Rated Event Formatter from the remote production sites to the primary production site by using the SFTP utility.

Failing Over to a Backup Site

You enable a backup or remote production site to take over the role of the primary production site. After the original production site is fixed, you can return the sites to their original state by switching the workload back to the original production site.

To fail over to a backup site:

1. On the backup site, stop replicating the ECE cache data to the primary production site by running the following command:

   ```
   gridSync stop PrimaryProductionClusterName
   ```

   where *PrimaryProductionClusterName* is the name of the cluster in the primary production site.

2. On the backup site, do the following:

   a. Change the BRM, PDC, and Customer Updater connection details to connect to BRM and PDC on the backup site by using a JMX editor.

   > **Note:**
   >
   > If only ECE in the primary production site failed and BRM and PDC in the primary production site are still running, you need not change the BRM and PDC connection details on the backup site.

   b. Start BRM and PDC.

3. Recover the data in the Oracle NoSQL database data store of the primary production site by performing the following:

   a. Convert the secondary Oracle NoSQL database data store node of the primary production site to the primary Oracle NoSQL database data store node by performing

a failover operation in the Oracle NoSQL database data store. For more information, see "Performing a Failover" in *Oracle NoSQL Database Administrator's Guide*.
The secondary Oracle NoSQL database data store node of the primary production site is now the primary Oracle NoSQL database data store node of the primary production site.

b. On the backup site, convert the rated events from the Oracle NoSQL database data store node that you just converted into the primary node into CDR files by starting Rated Event Formatter.

c. In a backup or remote production site, load the CDR files that you just converted into BRM by using Rated Event (RE) Loader.

d. Shut down the Oracle NoSQL database data store node that you just converted into the primary node.
See the "stop" utility in *Oracle NoSQL Database Administrator's Guide* for more information.

e. Stop the Rated Event Formatter that you just started.

4. In a backup or remote production site, start Pricing Updater, Customer Updater, and EM Gateway by running the following commands:

```
start pricingUpdater
start customerUpdater
start emGateway
```

All pricing and customer data is now back in the ECE grid in the backup or remote production site.

5. Stop and restart BRM Gateway.

6. Migrate internal BRM notifications from the primary production site to a backup or remote production site. See "Migrating ECE Notifications" for more information.

> **Note:**
>
> - If the expiry duration is configured for these notifications, ensure that you migrate the notifications before they expire. For the expiry duration, see the **expiry-delay** entry for the ServiceContext module in the *ECE_home***/config/charging-cache-config.xml** file.
> - All external notifications from a production site are published to the respective JMS queue. Diameter Gateway retrieves the notifications from the JMS queue and replicates to other sites based on the configuration.

7. Ensure that the network clients route all requests to the backup or remote production site.

Switching Back to the Original Production Site

You restart the restored production site and run the **gridSync** utility to restore the data.

To switch back to the original production site:

1. Install ECE and other required components in the original primary production site. For more information, see "Installing Elastic Charging Engine" in *ECE Installation Guide*.

   > **Note:**
   >
   > If only ECE in the original primary production site failed and BRM and PDC in the original primary production site are still running, install only ECE and provide the connection details about BRM and PDC in the original primary production site during ECE installation.

2. Configure primary and secondary Oracle NoSQL data store nodes. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".
3. On the machine on which the Oracle WebLogic server is installed, verify that the JMS queues have been created for loading pricing data and for sending event notification, and that JMS credentials have been configured correctly.
4. Set the following parameter in the *ECE_home***/config/ece.properties** file to **false**:

   ```
   loadConfigSettings = false
   ```

   The configuration data is not loaded in memory.
5. Add all details about participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging- coherence-override-prod.xml**).

   To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file. Table 75-2 provides the federation configuration parameter descriptions and default values.
6. Go to the *ECE_home***/bin** directory.
7. Start ECC:

   ```
   ./ecc
   ```
8. Start the charging server nodes:

   ```
   start server
   ```
9. Replicate the ECE cache data to the original production site by using the **gridSync** utility. For more information, see "Replicating ECE Cache Data".
10. Start the following ECE processes and gateways:

    ```
    start brmGateway
    start ratedEventFormatter
    start diameterGateway
    start radiusGateway
    start httpGateway
    ```
11. Verify that the same number of entries as in the new production site are available in the customer, balance, configuration, and pricing caches in the original production site by using the **query.sh** utility.
12. Stop Pricing Updater, Customer Updater, and EM Gateway in the new primary production site and then start them in the original primary production site.
13. Migrate internal BRM notifications from the new primary production site to the original primary production site. For more information, see "Migrating ECE Notifications".

14. Change the BRM Gateway, Customer Updater, and Pricing Updater connection details to connect to BRM and PDC in the original primary production site by using a JMX editor.
15. Stop RE Loader in the new primary production site and then start it in the original primary production site.
16. Stop and restart BRM Gateway in both the new primary production site and the original primary production site.

    The roles of the sites are now reversed to the original roles.

About the Segmented Active-Active System

The segmented active-active configuration consists of two active production sites at different geographic locations. This system uses one primary production site and one or more remote production sites. All sites concurrently process ECE requests for a different set of customers. ECE requests are routed across the sites based on your load balancing configuration. When any of the production sites do not respond, the requests from that site are diverted to the other sites. This configuration supports a minimal amount of downtime.

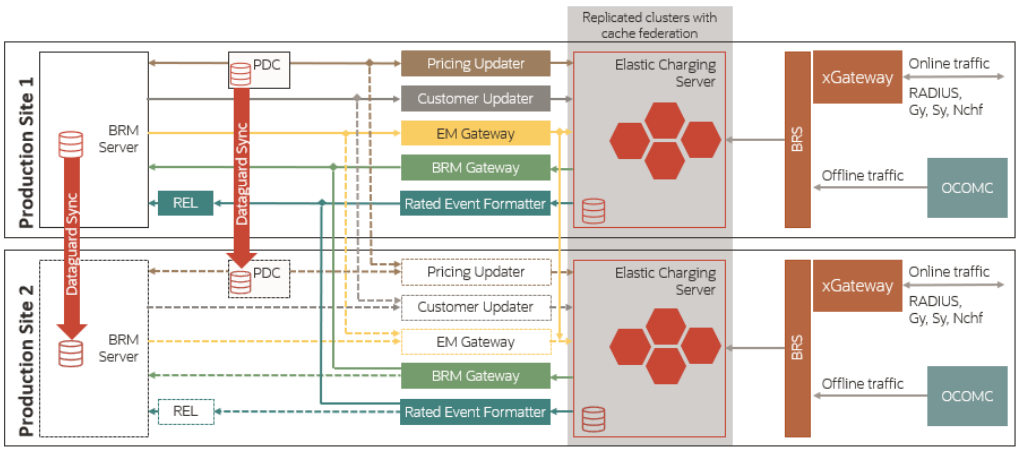
In the segmented active-active configuration, the backup system is started and processing usage requests. However, as with an active-active configuration, Customer Updater, Pricing Updater, and EM Gateway are configured, but not started. All changes to customer and pricing data are made on the production system and replicated on the backup system. You also need to configure load balancing to route usage requests to the production and backup systems.

To switch to the backup system, you configure all usage requests to go to it. You also start Customer Updater, Pricing Updater, and EM Gateway. These processes are connected to the BRM and PDC systems on the backup system.

You use the Secure Shell (SSH) File Transfer Protocol (SFTP) utility to replicate CDR files generated by Rated Event Formatter between participant sites.

Figure 75-4 shows the architecture for a segmented active-active disaster recovery system.

Figure 75-4 Architecture of Segmented Active-Active Disaster Recovery System



A few key aspects of the architecture are:

- The production site and backup site are active, and process requests. However, the network routes the requests for a subscriber participating in a sharing group to the same site, as the sharing group owner.
- ECE available on any site is configured to replicate the data with ECE clusters in other sites using Coherence federation. All ECE clusters are configured to work with BRM instance on a particular active site.

If you are using the Oracle NoSQL database for storing rated events, the data in the Oracle NoSQL database data store is replicated between the active sites. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".

About Load Balancing in a Segmented Active-Active System

In a segmented active-active system, ECE requests are routed across the sites based on your load balancing configuration. To ensure proper load balancing on your system, you can use a combination of global and local load balancers. The local load balancer routes the connection requests across the full range of available Diameter Gateway, RADIUS Gateway, and HTTP Gateway nodes. The global load balancer routes the connection requests to the gateway nodes on only one site, unless it detects that site is busy or if the local load balancer signals that it cannot reach ECE. You can set up your own load balancing configuration based on your requirements.

Configuring a Segmented Active-Active System

When configuring the segmented active-active system with ECE persistence enabled, consider the following points:

- Do not set the **loadConfigSettings** parameter in the secondary site, as you need to load **appConfig** on both sites. For more information, see "Specifying Driver Machine Properties" in *ECE Installation Guide*.
- Synchronize the wallet correctly on both sites. Whenever passwords are updated, you need to copy the wallet manually to the other site.
- Clean the persistence store when you are restarting the system. Whenever a site fails to respond and if you want to bring up the site using the **gridsync** utility from the other site, you must:
  - Start the site from a clean state in the same way the secondary cluster started.
  - Clear old data from the persistence store.
- Run only one **RatedEventFormatter** instance among the sites for a given RatedEventFormatter configuration.

To configure a segmented active-active system:

1. In the primary production site, do the following:
   a. Configure primary and secondary Oracle NoSQL database data store nodes. See "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide* for more information.
   b. Configure the ECE components (Customer Updater, EM Gateway, and so on).
   c. Add details about all participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging-coherence-override-prod.xml**).

To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file.

Table 75-2 provides the federation configuration parameter descriptions and default values.

d. Start ECE. See "Starting ECE" for more information.

2. On the backup site, do the following:

a. Configure primary and secondary Oracle NoSQL database data store nodes. See "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide* for more information.

b. Configure the ECE components (Customer Updater, EM Gateway, and so on).

> **Note:**
>
> - The name of Diameter Gateway, RADIUS Gateway, HTTP Gateway, Rated Event Formatter, and Rated Event Publisher for each site is unique.
> - Minimum two instances of Rated Event Formatter are configured to allow for failover.

c. Set the following parameter in the *ECE_home***/config/ece.properties** file to **false**:

```
loadConfigSettings = false
```

The application-configuration data is not loaded into memory when you start the charging server nodes.

d. Add details about all participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging-coherence-override-prod.xml**).

To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file. Table 75-2 provides the federation configuration parameter descriptions and default values.

e. Start the Elastic Charging Controller (ECC):

```
./ecc
```

f. Start the charging server nodes:

```
start server
```

3. On the primary production site, run the following commands:

```
gridSync start
gridSync replicate
```

The federation service is started and all existing data is replicated to the backup or remote production sites.

4. On the backup sites, do the following:

a. Verify that the same number of entries as in the primary production site are available in the customer, balance, configuration, and pricing caches in the backup or remote production sites by using the **query.sh** utility.

b. Verify that the charging server nodes in the backup or remote production sites are in the same state as the charging server nodes in the primary production site.

c. Configure the following ECE components by using a JMX editor:

- Rated Event Formatter
- Rated Event Publisher
- Diameter Gateway
- RADIUS Gateway
- HTTP Gateway

> **Note:**
>
> Ensure the following:
>
> - The name of Diameter Gateway, RADIUS Gateway, HTTP Gateway, Rated Event Formatter, and Rated Event Publisher for each site is unique.
> - A minimum two instances of Rated Event Formatter are configured to allow for failover.

d. Start the following ECE processes and gateways:

```
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
start httpGateway
```

The remote production sites are up and running with all required data.

e. Run the following command:

```
gridSync start
```

The federation service is started to replicate the data from the backup or remote production sites to the primary production site.

After starting Rated Event Formatter in the remote production sites, ensure that you copy the CDR files generated by Rated Event Formatter from the remote production sites to the primary production site by using the SFTP utility.

Failing Over to a Backup Site

You enable a backup or remote production site to take over the role of the primary production site. After the original production site is fixed, you can return the sites to their original state by switching the workload back to the original production site.

To fail over to a backup site:

1. On the backup site, stop replicating the ECE cache data to the primary production site by running the following command:

```
gridSync stop PrimaryProductionClusterName
```

where *PrimaryProductionClusterName* is the name of the cluster in the primary production site.

2. On the backup site, do the following:
   a. Change the BRM, PDC, and Customer Updater connection details to connect to BRM and PDC on the backup site by using a JMX editor.

      > **Note:**
      >
      > If only ECE in the primary production site failed and BRM and PDC in the primary production site are still running, you need not change the BRM and PDC connection details on the backup site.

   b. Start BRM and PDC.
3. Recover the data in the Oracle NoSQL database data store of the primary production site by performing the following:
   a. Convert the secondary Oracle NoSQL database data store node of the primary production site to the primary Oracle NoSQL database data store node by performing a failover operation in the Oracle NoSQL database data store. For more information, see "Performing a Failover" in *Oracle NoSQL Database Administrator's Guide*.
      The secondary Oracle NoSQL database data store node of the primary production site is now the primary Oracle NoSQL database data store node of the primary production site.
   b. On the backup site, convert the rated events from the Oracle NoSQL database data store node that you just converted into the primary node into CDR files by starting Rated Event Formatter.
   c. In a backup or remote production site, load the CDR files that you just converted into BRM by using Rated Event (RE) Loader.
   d. Shut down the Oracle NoSQL database data store node that you just converted into the primary node.
      See the "stop" utility in *Oracle NoSQL Database Administrator's Guide* for more information.
   e. Stop the Rated Event Formatter that you just started.
4. In a backup or remote production site, start Pricing Updater, Customer Updater, and EM Gateway by running the following commands:

   ```
   start pricingUpdater
   start customerUpdater
   start emGateway
   ```

   All pricing and customer data is now back in the ECE grid in the backup or remote production site.
5. Stop and restart BRM Gateway.
6. Migrate internal BRM notifications from the primary production site to a backup or remote production site. See "Migrating ECE Notifications" for more information.

> **Note:**
>
> - If the expiry duration is configured for these notifications, ensure that you migrate the notifications before they expire. For the expiry duration, see the **expiry-delay** entry for the ServiceContext module in the *ECE_home***/config/charging-cache-config.xml** file.
> - All external notifications from a production site are published to the respective JMS queue. Diameter Gateway retrieves the notifications from the JMS queue and replicates to other sites based on the configuration.

7. Ensure that the network clients route all requests to the backup or remote production site.

Switching Back to the Original Production Site

You restart the restored production site and run the **gridSync** utility to restore the data.

To switch back to the original production site:

1. Install ECE and other required components in the original primary production site. For more information, see "Installing Elastic Charging Engine" in *ECE Installation Guide*.

   > **Note:**
   >
   > If only ECE in the original primary production site failed and BRM and PDC in the original primary production site are still running, install only ECE and provide the connection details about BRM and PDC in the original primary production site during ECE installation.

2. Configure primary and secondary Oracle NoSQL data store nodes. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".
3. On the machine on which the Oracle WebLogic server is installed, verify that the JMS queues have been created for loading pricing data and for sending event notification, and that JMS credentials have been configured correctly.
4. Set the following parameter in the *ECE_home***/config/ece.properties** file to **false**:

   ```
   loadConfigSettings = false
   ```

   The configuration data is not loaded in memory.
5. Add all details about participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging- coherence-override-prod.xml**).

   To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file. Table 75-2 provides the federation configuration parameter descriptions and default values.
6. Go to the *ECE_home***/bin** directory.
7. Start ECC:

   ```
   ./ecc
   ```

8. Start the charging server nodes:

```
start server
```

9. Replicate the ECE cache data to the original production site by using the **gridSync** utility. For more information, see "Replicating ECE Cache Data".

10. Start the following ECE processes and gateways:

```
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
start httpGateway
```

11. Verify that the same number of entries as in the new production site are available in the customer, balance, configuration, and pricing caches in the original production site by using the **query.sh** utility.

12. Stop Pricing Updater, Customer Updater, and EM Gateway in the new primary production site and then start them in the original primary production site.

13. Migrate internal BRM notifications from the new primary production site to the original primary production site. For more information, see "Migrating ECE Notifications".

14. Change the BRM Gateway, Customer Updater, and Pricing Updater connection details to connect to BRM and PDC in the original primary production site by using a JMX editor.

15. Stop RE Loader in the new primary production site and then start it in the original primary production site.

16. Stop and restart BRM Gateway in both the new primary production site and the original primary production site.

    The roles of the sites are now reversed to the original roles.

About the Active-Active System

The active-active configuration consists of two active production sites at different geographic locations. All sites concurrently process the ECE requests based on the customerGroup list. ECE requests are routed across the sites based on your load-balancing configuration. All updates occurring in a site's ECE cluster are replicated to other sites using the Coherence cache federation.

When a production site does not respond, the request from that site is diverted to the other backup sites. This configuration supports the least amount of downtime.

In the active-active configuration, you configure Customer Updater, Pricing Updater, and EM Gateway in the production and backup systems. All customer and pricing data changes are made on the production system and then replicated on the backup system.

Each subscriber in the system is assigned to a customerGroup list. ECE distributes customers in the customerGroup list based on the CutomerGroupconfiguration parameters. The customerGroup list helps maintain a balanced workload across the clusters in a two-site ECE deployment. If a primary production site does not respond, the request is routed to the backup production site. Also, you must configure load balancing to route the usage requests to production and backup systems.

> **Note:**
>
> To ensure that all sharing group members are in the same customerGroup, assign each account to only one sharing group.

To switch to the backup system, you use Monitor Agent. See "Configuring an Active-Active System" for more information. This action diverts the traffic from a preferred site to the next backup site. When you indicate that the preferred site has started functioning again, the traffic is routed back to the preferred site.

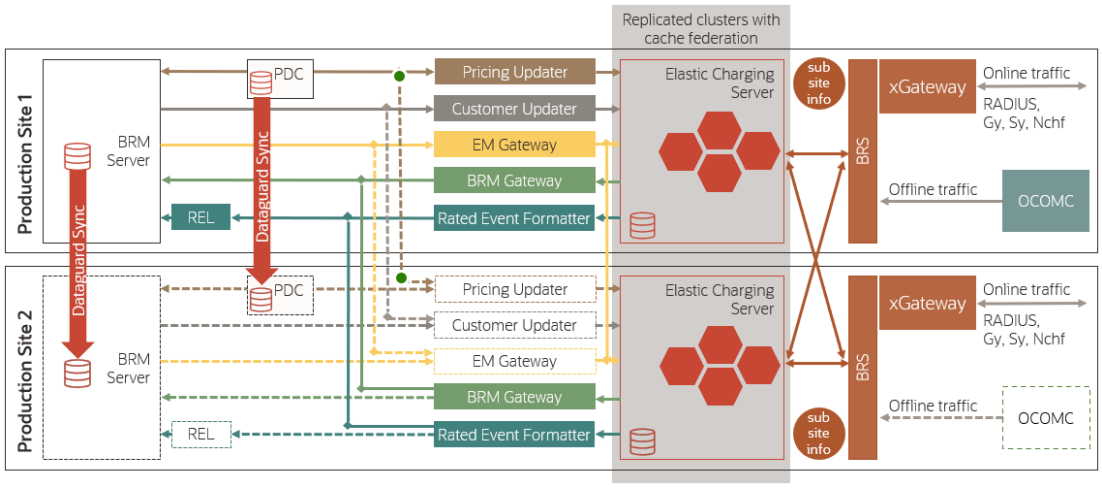
You can configure the ECE active-active mode to process usage requests in the site that receives the usage request, irrespective of the subscriber's preferred site. For example, if production site 1 receives a subscriber's request, it is processed in production site 1. Similarly, if production site 2 receives the usage request, it is processed in production site 2. See "Processing Usage Requests in the Site Received" for more information.

> **Note:**
>
> This configuration does not apply to usage charging requests for members of a sharing group. Usage requests for sharing group members continue to be processed on the same site as the sharing group parent.

You use the Secure Shell (SSH) File Transfer Protocol (SFTP) utility to replicate CDR files generated by Rated Event Formatter between participant sites.

Figure 75-5 shows the architecture for an active-active disaster recovery system.

Figure 75-5 Architecture of Active-Active Disaster Recovery System



A few critical aspects of the architecture are:

- Two sites are supported in the active-active mode, and both sites are active.
- BatchRequestService (BRS) ensures intelligent routing to the appropriate site if the site that receives the request is not a subscriber's preferred site.

- ECE available in any site is configured to replicate data with ECE clusters in other sites using Coherence federation. All ECE clusters are configured to work with a BRM instance on a particular active site.

If you are using the Oracle NoSQL database for storing rated events, the data in the data store is replicated between the active sites. For more information, see "About Configuring Oracle NoSQL Database Data Store Nodes".

About Load Balancing in an Active-Active System

In an active-active system, EM Gateway routes BRM update requests across sites based on the app and site configurations to ensure load balancing.

EM Gateway routes connection requests to Diameter Gateway, RADIUS Gateway, and HTTP Gateway nodes in one of the active sites. If the site does not respond, the request is rerouted to the backup production site.

You can set up load balancing configuration based on your requirements.

About Rated Event Formatter in a Persistence-Enabled Active-Active System

When data persistence is enabled, each site in an active-active system has a primary Rated Event Formatter instance for each schema, and at least one secondary instance for each schema.

As rated events are created, the following happens on each site:

1. ECE creates rated events and commits them to the Coherence cache. Each rated event created by ECE includes the Coherence cluster name of the site where it was created.
2. The Coherence federation service replicates the events to the remote sites, as it does for other federated objects.
3. Coherence caching persists the events to the database in batches. Each schema at each site has its own rated event database table.
4. The primary Rated Event Formatter instance processes all rated events from the corresponding site-specific database table.
5. The primary Rated Event Formatter instance commits the formatted events to the cache as a checkpoint. The site name is included in the checkpoint data, along with the schema number, timestamp, and plugin type.
6. The Coherence federation service replicates the checkpoint to the remote sites, as it does for other federated objects. The remote site ECE servers then purge the events persisted in the checkpoint from the database in batches by schema and by site.
7. Coherence caching persists the checkpoint to the database to be consumed by Rated Event Loader. Checkpoints are grouped by schema and by site.
8. The ECE server purges the events related to the persisted checkpoint. Events are purged from the database in batches by schema and by site.

Remote sites that receive federated events and checkpoints similarly persist them to and purge them from the database, in site and schema-specific database tables. In this way, all sites contain the same rated events and checkpoints, no matter where

they were generated, and each rated event and checkpoint retains information about the site that generated it. If the Rated Event Formatter instance at any one site is down, a secondary instance at a remote site can immediately begin processing the rated events, preserving the site-specific information as though it were the original site. See "Resolving Rated Event Formatter Instance Outages".

Resolving Rated Event Formatter Instance Outages

If a primary Rated Event Formatter instance is down, take one of the following approaches, depending on whether the outage is planned or unplanned, and considering your operational needs:

- **Planned outage: Primary instance finishes processing**: Choose this option for planned outages, when rating stops but the primary Rated Event Formatter instance can keep processing.
  1. After no new rated events are being generated by the site, wait until the local Rated Event Formatter has finished processing all rated events from the site.
  2. In the remote sites, drop or truncate the rated event database table for the rated events federated from the site with the outage. Dropping the table means you must recreate it and its indexes after resolving the outage.
  3. Stop the Rated Event Formatter at the site with the outage.
  4. When the outage is resolved, you can start Rated Event Formatter again to resume processing events.
- **Unplanned outage: Secondary instance takes over processing**: Choose this option for unplanned outages, when the primary Rated Event Formatter is also down. After failing over to the backup site as described in "Failing Over to a Backup Site (Active-Active)", perform the following tasks:
  1. Confirm that the last successful Rated Event Formatter checkpoint for the local site matches the one federated to the remote site. You can use the JMX queryRatedEventCheckPoint operation in the ECE configuration MBeans. See "Getting Rated Event Formatter Checkpoint Information".
  2. If needed, start the secondary Rated Event Formatter instance on the remote site.
  3. Activate the secondary Rated Event Formatter instance on the remote site using the JMX activateSecondaryInstance operation in the ECE monitoring MBeans. See "Activating a Secondary Rated Event Formatter Instance".
     The secondary instance takes over processing the federated rated events as though it were the primary instance at the site with the outage. The events and checkpoints are persisted in the database tables for the original site, not the remote site.
  4. Wait until the secondary instance has finished processing all rated events federated from the site with the outage.
  5. At the site with the outage, drop or truncate the rated event database table for local events. Dropping the table means you must recreate it and its indexes after resolving the outage.
  6. Stop the secondary Rated Event Formatter instance.
  7. When the outage is resolved and the site has been recovered as described in "Switching Back to the Original Production Site (Active-Active)", restart the primary Rated Event Formatter again to resume processing events at the local site. If you had the secondary Rated Event Formatter instance running at the remote site before the outage, restart it too.

Getting Rated Event Formatter Checkpoint Information

You can retrieve information about the last Rated Event Formatter checkpoint committed to the database.

To retrieve information about the last Rated Event Formatter checkpoint:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand the database connection you want checkpoint information from.
4. Expand **Operations**.
5. Run the **queryRatedEventCheckPoint** operation.
   Checkpoint information appears for all Rated Event Formatter instances using the database connection. Information includes site, schema, and plugin names as well as the time of the most recent checkpoint.

Activating a Secondary Rated Event Formatter Instance

If a primary Rated Event Formatter instance is down, you can activate a secondary instance to take over rated event processing.

To activate a secondary Rated Event Formatter instance:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Monitoring** node.
3. Expand **RatedEventFormatterMatrices**.
4. Expand **Operations**.
5. Run the **activateSecondaryInstance** operation.
   The secondary Rated Event Formatter instance begins processing rated events.

About CDR Generator in an Active-Active System

When CDR generation is enabled, each site in an active-active system contains a CDR Generator, and each site can generate unrated CDRs for external systems. For information about CDR Generator, see "Generating CDRs" in *ECE Implementing Charging*.

When a production site goes down, the CDR database retains all in-progress (or incomplete) CDR sessions, and all unrated 5G usage events are diverted to the CDR Gateway on the other production site.

When the failed production site comes back online, the CDR Formatter finds and purges all incomplete CDR sessions from the CDR database that are older than a configurable duration. For example, if it is 12:00:00 and the configurable duration is 200 seconds, the CDR Formatter would purge from the CDR database all incomplete CDRs that were last updated today at 11:56:40 or earlier. You use the CDR Formatter's **cdrOrphanRecordCleanupAgeInSec** attribute to set the configurable duration. See "Configuring the CDR Formatter" in *ECE Implementing Charging*.

Configuring an Active-Active System

> **Note:**
>
> Active-active disaster recovery configurations are supported only by ECE 12.0.0.3.0 with Interim Patch 31848507 and later. The Interim Patch includes an SDK to help you migrate ECE to 12.0.0.3.0 Interim Patch 31848507 so you can use the active-active configuration. See the Interim Patch README for more information.

To configure an active-active system:

1. In the primary production site, do the following:
   a. Configure primary and secondary Oracle NoSQL database data store nodes. See "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide* for more information.
   b. Configure the ECE components (Customer Updater, EM Gateway, and so on).
   c. Add all details about participant sites to the **federation-config** section of the ECE Coherence override file (for example, *ECE_home***/config/charging-coherence-override-prod.xml**).
      To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_home***/config/ece.properties** file.

      See Table 75-2 for more information about providing the federation configuration parameter descriptions and default values.
   d. Go to the *ECE_home*/**config/management** directory, where *ECE_home* is the directory in which ECE is installed.
   e. Configure HTTP Gateway. See "Connecting ECE to a 5G Client" in *ECE Implementing Charging* for more information.
   f. Open the **charging-settings.xml** file.
   g. In the **CustomerGroupConfiguration** section, set the app configuration parameters as shown in the following sample file:

```
<customerGroupConfigurations config-
class="oracle.communication.brm.charging.appconfiguration.beans.customergroup.C
ustomerGroupConfigurations">
               <customerGroupConfigurationList>
                    <customerGroupConfiguration
                         config-
class="oracle.communication.brm.charging.appconfiguration.beans.customergroup.C
ustomerGroupConfiguration" name="customerGroup5">
                         <clusterPreferenceList.name config-
class="java.util.ArrayList">

                     <clusterPreferenceConfiguration
                         config-
class="oracle.communication.brm.charging.appconfiguration.beans.customergroup.C
lusterPreferenceConfiguration"
                         name="BRM-S2"
                         priority="1" routingGatewayList="host1:port1"/>
                         </clusterPreferenceList>
```

```
                    </customerGroupConfiguration>
                    <customerGroupConfiguration
                         config-
class="oracle.communication.brm.charging.appconfiguration.beans.customerg
roup.CustomerGroupConfiguration" name="customerGroup2">
                         <clusterPreferenceList config-
class="java.util.ArrayList">

                         <clusterPreferenceConfiguration
                         config-
class="oracle.communication.brm.charging.appconfiguration.beans.customerg
roup.ClusterPreferenceConfiguration"
                         name="BRM-S2"
                         priority="1"
routingGatewayList="host1:port1,host1:port1"/>


                         <clusterPreferenceConfiguration
                         config-
class="oracle.communication.brm.charging.appconfiguration.beans.customerg
roup.ClusterPreferenceConfiguration"
                         name="BRM-S1"
                         priority="2"
routingGatewayList="host2:port2,host2:port2"/>
                         </clusterPreferenceList>
                    </customerGroupConfiguration>
               </customerGroupConfigurationList>
          </customerGroupConfigurations>
```

Table 75-3 provides the configuration parameters of the **CustomerGroupConfiguration** section.

Table 75-3 CustomerGroupConfiguration Parameters

Configuration	Parameter and Description
CustomerGroupConfiguration	• `name=` Customers are processed and distributed in active-active system sites based on **customerGroup**. The customer names configured in **customerGroup** are updated to the PublicUserIdentity (PUI) cache when you load the customer information to ECE through **customerUpdater** or when you create or update information of customers in BRM using EM Gateway. • `clusterPreferenceList=` Includes a list of cluster names and priority for each cluster name for routing the requests during a site failure.

Table 75-3 (Cont.) CustomerGroupConfiguration Parameters

Configuration	Parameter and Description
clusterPreferenceConfiguration	• `name=` Name of the cluster. • `clusterPreferenceConfiguration.priority=` The priority of the preferred cluster that is assigned in the customerGroup list to process the rating request. The priority to process the request is in the incremental order of numbers and assigned to the lowest number. For example, if you set the value to 1 for priority, the cluster associated with this number processes the request first. • `routingGatewayList=` A comma-separated list of the host name and port number of chargingServer values used for httpGateway.

h. If data persistence is enabled, configure a primary and secondary Rated Event Formatter instance for each site in the **ratedEventFormatter** section, as shown in the following sample file:

```
<ratedEventFormatterConfigurationList config-class="java.util.ArrayList">
    <ratedEventFormatterConfiguration
            config-
class="oracle.communication.brm.charging.appconfiguration.beans.ratedeventforma
tter.RatedEventFormatterConfiguration"
            name="ref_site1_primary"
            partition="1"
            connectionName="oracle1"
            siteName="site1"
            threadPoolSize="2"
            retainDuration="0"
            ripeDuration="30"
            checkPointInterval="20"
            maxPersistenceCatchupTime="0"
            pluginPath="ece-ratedeventformatter.jar"

pluginType="oracle.communication.brm.charging.ratedevent.formatterplugin.intern
al.SampleFormatterPlugInImpl"
            pluginName="brmCdrPluginDC1Primary"
            noSQLBatchSize="25" />
    <ratedEventFormatterConfiguration
            config-
class="oracle.communication.brm.charging.appconfiguration.beans.ratedeventforma
tter.RatedEventFormatterConfiguration"
            name="ref_site1_secondary"
            partition="1"
            connectionName="oracle2"
            siteName="site1"
            primaryInstanceName="ref_site1_primary"
            threadPoolSize="2"
            retainDuration="0"
            ripeDuration="30"
            checkPointInterval="20"
            maxPersistenceCatchupTime="0"
            pluginPath="ece-ratedeventformatter.jar"

pluginType="oracle.communication.brm.charging.ratedevent.formatterplugin.intern
al.SampleFormatterPlugInImpl"
            pluginName="brmCdrPluginDC1Secondary"
```

```
            noSQLBatchSize="25" />
    <ratedEventFormatterConfiguration
            config-
class="oracle.communication.brm.charging.appconfiguration.beans.ratedeven
tformatter.RatedEventFormatterConfiguration"
            name="ref_site2_primary"
            partition="1"
            connectionName="oracle2"
            siteName="site2"
            threadPoolSize="2"
            retainDuration="0"
            ripeDuration="30"
            checkPointInterval="20"
            maxPersistenceCatchupTime="0"
            pluginPath="ece-ratedeventformatter.jar"

pluginType="oracle.communication.brm.charging.ratedevent.formatterplugin.
internal.SampleFormatterPlugInImpl"
            pluginName="brmCdrPluginDC2Primary"
            noSQLBatchSize="25" />
    <ratedEventFormatterConfiguration
            config-
class="oracle.communication.brm.charging.appconfiguration.beans.ratedeven
tformatter.RatedEventFormatterConfiguration"
            name="ref_site2_secondary"
            partition="1"
            connectionName="oracle1"
            siteName="site2"
            primaryInstanceName="ref_site2_primary"
            threadPoolSize="2"
            retainDuration="0"
            ripeDuration="30"
            checkPointInterval="20"
            maxPersistenceCatchupTime="0"
            pluginPath="ece-ratedeventformatter.jar"

pluginType="oracle.communication.brm.charging.ratedevent.formatterplugin.
internal.SampleFormatterPlugInImpl"
            pluginName="brmCdrPluginDC2Secondary"
            noSQLBatchSize="25" />
</ratedEventFormatterConfigurationList>
```

The **siteName** property determines the site that the instance processes rated events for. This lets you configure secondary instances as backups for remote sites. The sample specifies that the **ref_site1_secondary** instance running is running at site 2, but processes rated events federated from site 1 in case of an outage.

i. Configure the production sites to process the routing requests.

j. Open the **site-configuration.xml** file.

Configure all **monitorAgent** instances from all sites. Each Monitor Agent instance includes the Coherence cluster name, host name or IP address, and JMX port.

Table 75-4 provides the configuration parameters of Monitoring Agent.

Table 75-4 Monitor Agent Configuration Parameters

Name	Description
name	The name of the production or remote site where the request should be processed. These should correspond to site names defined for the Rated Event Formatter instances.
host	The IP address of the participant site.
jmxPort	**jmxPort** of the production or remote site.
disableMonitor	This configuration allows a **monitorAgent** instance to disable collecting monitoring results from multiple **monitorAgent** instances running within a site. It prevents generating redundant monitoring results for a site. **Note**: Default value is set to **false**. If you set this value to **true**, **monitorAgent** instance disallows collecting redundant monitoring results.

> **Note:**
>
> The **monitorAgent** properties should match with the properties in the **eceTopology.conf** file where a **monitorAgent** instance is configured to start from a specific production site.

k. Copy the **JMSConfiguration.xml** file content of all sites to a single file and enter the following details:

- Add the `<Cluster>clusterName</Cluster>` tag for the queue types.
- Import the wallet for all clusters and specify the wallet path in the `<KeyStoreLocation> and <ECEWalletLocation>` locations.

l. In the **eceTopology.conf** file, enable the JMX port for all ECS server nodes and clients, such as Diameter Gateway, HTTP Gateway, RADIUS Gateway, and EM Gateway. Also, enable the JMX port for each Monitor Agent instance.

m. Start ECE. See "Starting ECE" for more information.

2. On the backup or remote site, do the following:

a. Configure primary and secondary Oracle NoSQL database data store nodes. For more information, see "Configuring the KVStore" in *Oracle NoSQL Database Administrator's Guide*.

b. Configure the ECE components (Customer Updater, EM Gateway, and so on).

Ensure the following:

- The name of Diameter Gateway, RADIUS Gateway, HTTP Gateway, Rated Event Formatter, and Rated Event Publisher for each site is unique.
- At least two instances of Rated Event Formatter are configured to allow for failover. For a data persistence-enabled system, configure at least one primary and one secondary instance for each site.

c. Set the following parameter in the *ECE_home*/**config/ece.properties** file to **false**:

```
loadConfigSettings = false
```

The application-configuration data is not loaded into memory when you start the charging server nodes.

d. Add all the details of participant sites in the federation-config section of the ECE Coherence override file (for example, *ECE_home***/config/charging-coherence-override-prod.xml**).

   To confirm which ECE Coherence override file is used, see the **tangosol.coherence.override** value in the *ECE_ home***/config/ ece.properties** file. Table 75-2 provides the federation configuration parameter descriptions and default values.

e. Start the Elastic Charging Controller (ECC):

```
./ecc
```

f. Start the charging server nodes:

```
start server
```

3. On the primary production site, run the following commands:

```
gridSync start
gridSync replicate
```

   The federation service is started and all the existing data is replicated to the backup or remote production sites.

4. On the backup sites, do the following:

   a. Verify that the same number of entries as in the primary production site are available in the customer, balance, configuration, and pricing caches in the backup or remote production sites by using the **query.sh** utility.

   b. Verify that the charging server nodes in the backup or remote production sites are in the same state as the charging server nodes in the primary production site.

   c. Configure the following ECE components and the Oracle NoSQL database connection details by using a JMX editor:

      - Rated Event Formatter
      - Rated Event Publisher
      - Diameter Gateway
      - RADIUS Gateway
      - HTTP Gateway

      Ensure the following:

      - The name of Diameter Gateway, RADIUS Gateway, HTTP Gateway, Rated Event Formatter, and Rated Event Publisher for each site is unique.
      - At least two instances of Rated Event Formatter are configured to allow for failover. For a data persistence-enabled system, configure at least one primary and one secondary instance for each site.

   d. Start the following ECE processes and gateways:

```
start brmGateway
start ratedEventFormatter
start diameterGateway
start radiusGateway
start httpGateway
```

The remote production sites are up and running with all required data.

e. Run the following command:

```
gridSync start
```

The federation service is started to replicate the data from the backup or remote production sites to the preferred production site.

After starting Rated Event Formatter in the remote production sites, ensure that you copy the CDR files generated by Rated Event Formatter from the remote production sites to the primary production site by using the SFTP utility.

Including Custom Clients in Your Active-Active Configuration

If your system includes a custom client application that calls the ECE API, you need to add the custom client to your active-active disaster recovery configuration. This enables the active-active system architecture to automatically route requests from your custom client to a backup site when a site failover occurs. To do so, you configure the custom client as an ECE Monitor Framework-compliant node in the ECE cluster.

To add a custom client to an active-active configuration:

1. Modify your custom client to use the ECE Monitor Framework:
   a. Add this import statement:

```
import
oracle.communication.brm.charging.monitor.framework.internal.MonitorFramework;
```

   b. Add these lines to the program:

```
if (MonitorFramework.isJMXEnabledApp) {
   MonitorFramework monitorFramework = (MonitorFramework)
context.getBean(MonitorFramework.MONITOR_BEAN_NAME);
   try {
      monitorFramework.initializeMonitor(null); // null parameter for any non-
ECE Monitor Agent node
   }
   catch (Exception ex) {
      // Failed to initialize Monitor Framework, check log file
      System.exit(-1);
   }
}
...  // continue as before
```

2. When you start your custom client, include these Java system properties:
   - **-Dcom.sun.management.jmxremote.port** set to the port number for enabling JMX RMI connections. Ensure that you specify an unused port number.
   - **-Dcom.sun.management.jmxremote.rmi.port** set to the port number to which the RMI connector will be bound.
   - **-Dtangosol.coherence.member** set to the name of the custom client application instance running within the ECE cluster.

   For example:

```
java -Dcom.sun.management.jmxremote.port=6666 \
-Dcom.sun.management.jmxremote.rmi.port=6666 \
-Dtangosol.coherence.member=customApp1 \
-jar customApp1.jar
```

3. Edit the *ECE_home*/**config/eceTopology.conf** file to include a row for each custom client application instance. For each row, enter the following information:
   - **node-name**: The name of the JVM process for that node.
   - **role**: The role of the JVM process for that node.
   - **host name**: The host name of the physical server machine on which the node resides. For a standalone system, enter **localhost**.
   - **host ip**: If your host contains multiple IP addresses, enter the IP address so that Coherence can be pointed to a port.
   - **JMX port**: The JMX port of the JVM process for that node. By specifying a JMX port number for one node, you expose MBeans for setting performance-related properties and collecting statistics for all node processes. Enter any free port, such as 9999, for the charging server node to be the JMX-management enabled node.
   - **start CohMgt**: Specify whether you want the node to be JMX-management enabled.

   For example:

```
#node-name     |role       |host name  (no spaces!) |host ip  |JMX port  |
start CohMgt  |JVM Tuning File
customApp1     |customApp  |localhost               |         |6666      |
false         |
```

Including Offline Mediation Controller in Your Active-Active Configuration

If your system includes Oracle Communications Offline Mediation Controller, you need to add it to your active-active disaster recovery configuration. This enables the active-active system architecture to automatically route requests from Offline Mediation Controller to a backup site when a site failover occurs.

To include Offline Mediation Controller in your active-active configuration:

1. On each active production site, do the following:
   a. Log in to your ECE driver machine as the **rms** user.
   b. In your **ocecesdk/config/client-charging-context.xml** file, add the following line to the **beans** element:

```
<importresource="classpath:/META-INF/spring/monitor.framework-
context.xml"/>
```

2. On each Offline Mediation Controller machine, do the following:
   a. Log in to your Offline Mediation Controller machine as the **rms** user.
   b. Add the following lines to your *OCOMC_home*/**bin/nodemgr** file:

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.rmi.port=rmi_port
-Dcom.sun.management.jmxremote.port=port
```

   where:
   - *rmi_port* is set to the port number to which the RMI connector will be bound.

- *port* is set to the port number for enabling JMX RMI connections. Ensure that you specify an unused port number.

c. In the *OCOMC_home*/**bin/UDCEnvironment** file, set **JMX_ENABLED_STATUS** to **true** and set **JMX_PORT** to the desired JMX port number:

```
#For enabling jmx in an active-active setup
JMX_ENABLED_STATUS=true
JMX_PORT=9992
```

3. On each Offline Mediation Controller machine, restart Node Manager by going to the *OCOMC_home*/**bin** directory and running this command:

```
./nodemgr
```

Failing Over to a Backup Site (Active-Active)

To fail over to a backup site in an active-active configuration:

1. Open a JMX editor such as a JConsole.
2. Expand the **ECE Monitoring** node.
3. Expand **Agent**.
4. Expand **Operations**.
5. Set the **failoverSite()** operation to the name of the failed site.
6. Repeat the steps from 1 through 7 in the "Failing Over to a Backup Site " section. For Rated Event Formatter failover when data persistence is enabled, follow the steps in "Resolving Rated Event Formatter Instance Outages".

   The former backup site or one of the remote production sites is now the new preferred production site. When the preferred site starts functioning, you mark the recoverSite and the site traffic routes back to the preferred site. For more information, see "Switching Back to the Original Production Site (Active-Active)".

Switching Back to the Original Production Site (Active-Active)

To switch back to the original production site in an active-active system:

1. Repeat the steps 1 through 16 in the "Switching Back to the Original Production Site" section.
2. Open a JMX editor such as a JConsole.
3. Expand the **ECE Monitoring** node.
4. Expand **Agent**.
5. Expand **Operations**.
6. Set the **recoverSite()** operation to the name of the recovered site.
7. If data persistence is enabled and you failed over your Rated Event Formatter instance at the original site to a secondary instance at a remote site, restart any primary and secondary Rated Event Formatter instances at the original site.

Processing Usage Requests in the Site Received

To configure the ECE active-active mode to process usage requests in the site that receives the request irrespective of the subscriber's preferred site, perform the following steps:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.brsConfigurations.default**.
4. Expand **Attributes**.
5. Set the **skipActiveActivePreferredSiteRouting** attribute to **true**.

> **Note:**
>
> By default, the **skipActiveActivePreferredSiteRouting** attribute is set to **false**.

Replicating ECE Cache Data

In an active-hot standby system, a segmented active-active system, or an active-active system, when you configure or perform disaster recovery, you replicate the ECE cache data to the participant sites by using the **gridSync** utility.

To replicate the ECE cache data:

1. Go to the *ECE_home***/bin** directory.
2. Start ECC:

   ```
   ./ecc
   ```

3. Do one of the following:
   - To start replicating data to a specific participant site asynchronously and also replicate all the existing ECE cache data to a specific participant site, run the following commands:

     ```
     gridSync start [remoteClusterName]
     gridSync replicate [remoteClusterName]
     ```

     where *remoteClusterName* is the name of the cluster in a participant site.
   - To start replicating data to all the participant sites asynchronously and also replicate all the existing ECE cache data to all the participant sites, run the following commands:

     ```
     gridSync start
     gridSync replicate
     ```

See "gridSync" for more information on the **gridSync** utility.

Migrating ECE Notifications

When you failover to a backup site or switching back to the primary site, you must migrate the notifications to the destination site.

> **Note:**
>
> If you are using Apache Kafka for notification handling, notifications are not migrated to the destination site. Apache Kafka retains the notifications and these notifications appear in the original site or components when they are active.

To migrate ECE notifications:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **systemAdmin**.
4. Expand **Operations**.
5. Select **triggerFailedClusterServiceContextEventMigration**.
6. In the method's **failedClusterName** field, enter the name of the failed site's cluster.
7. Click the **triggerFailedClusterServiceContextEventMigration** button.

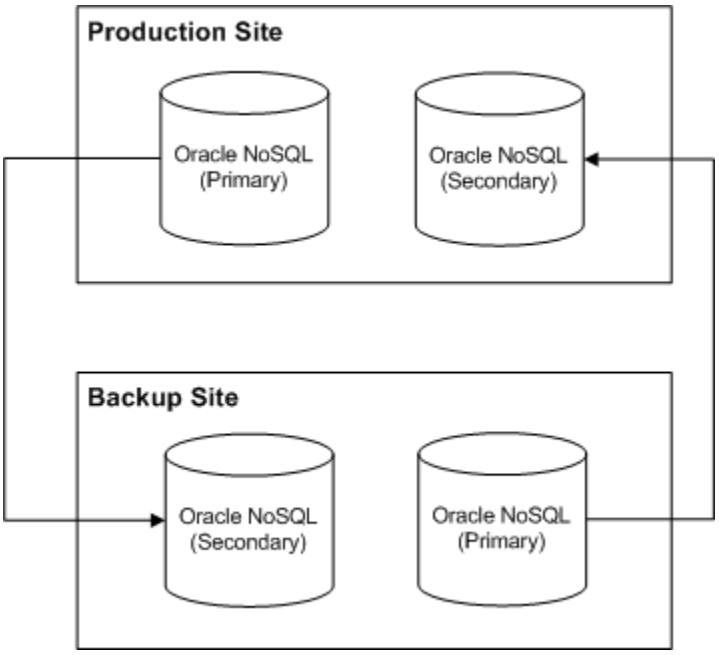
All the internal BRM notifications are migrated to the destination site. In an active-active system, the external notifications are also migrated to the destination site. If you cannot establish the WebLogic cluster subscription due to a site failover, you should restart Diameter Gateway on the destination site. If a site recovers from a failover, you should restart all the Diameter Gateway instances in the cluster.

About Configuring Oracle NoSQL Database Data Store Nodes

In an active-hot standby system, a segmented active-active system, or an active-active system, you configure primary and secondary storage nodes in the Oracle NoSQL database data store for each participant site to store and replicate the rated event data. The secondary storage nodes in one site act as a backup for the primary storage nodes of the other site. For example, when rated events are stored in the primary storage nodes of the production site, the information is replicated in the secondary storage nodes of the backup site. Similarly, the information that is stored in the primary storage nodes of the backup site is replicated in the secondary storage nodes of the production site.

Figure 75-6 shows the primary and secondary Oracle NoSQL database storage nodes configured in a disaster recovery system.

Figure 75-6 Primary and Secondary Oracle NoSQL Database Storage Nodes



Oracle recommends that you configure a minimum of three storage nodes in each site: two storage nodes to store rated events and one storage node to allow for failover. You can add additional storage nodes as needed to handle the expected throughput for your system. For information about adding and configuring storage nodes in the Oracle NoSQL database data store, see the Oracle NoSQL Database documentation.

76
Troubleshooting ECE

Learn how to troubleshoot Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE).

Topics in this document:

ECE Troubleshooting Checklist

Check the following to determine the state of the ECE system:

- Verify nodes are running by using a JMX editor such as JConsole.
- Verify the state machine state of nodes.

  To start processing requests, ECE goes through different states, such as startup, configuration, update, and processing and so on. A node that is in a state other than USAGE_PROCESSING will not process usage requests.
- Verify the charging-server health threshold.

  The number of running nodes may have gone below the threshold. See "Configuring the Charging-Server Health Threshold" for information.

Collecting Diagnostic Information

For collecting diagnostic information, turn on the ECC feedback mode which produces extra information when running commands. For example:

```
ecc:000>set feedback true
```

The feedback mode setting is saved in your local profile so you do not need to set it every time you start ECC.

Collecting Log Files for Sending to Oracle Technical Support

Use the **infoCollector** ECC command to collect log files from your ECE system. The command copies log files from all server machines and the driver machine and stores them in a TAR file, which you can send to Oracle technical support.

The **infoCollector** command does not collect files from clients such as Offline Mediation Controller.

To collect log files:

1. (Optional) Create a directory to put the collected log files.

   If you do not specify a directory, the command puts the collected log files in *ECE_home*.
2. Log on to the driver machine.
3. Go to *ECE_home***/bin** directory.
4. Run the following command:

   ```
   ./ecc
   ```
5. From the command prompt, run the **infoCollector** command.

   ```
   infoCollector
   ```

   See "infoCollector Syntax" for information about syntax and arguments for running the command.

   If you run the **infoCollector** command with no arguments, the following occurs:

   - From each server machine in your topology, the following file and directories are copied to the following location on the driver machine (where *user_home* is the user home directory of the driver machine and *server_host* is the IP address or host name of the server machine as it is defined in the *ECE_home***/config/eceTopology.conf** file):
     - *user_home***/***server_host***/VERSION**
     - *user_home***/***server_host***/logs**
     - *user_home***/***server_host***/config**
     - *user_home***/***server_host***/brm_config**
     - *user_home***/***server_host***/odi_transformation**
   - From the driver machine, the following file and directories are copied to the following location on the driver machine (where *user_home* is the user home directory of the driver machine and *driver_host* is the IP address or host name of the driver machine as it is defined in the *ECE_home***/config/eceTopology.conf** file):
     - *user_home***/***driver_host***/VERSION**
     - *user_home***/***driver_host***/config**
     - *user_home***/***driver_host***/brm_config**
   - A compressed TAR file is created with the extension **tar.gz** in the user home directory of the driver machine (for example, *user_home***/info_collector.tar.gz**).

For ECE processes started by running the **ecc** command, the GC debug logs are enabled by default. The GC debug log files are stored in the ECE logs directory (*ECE_home***/logs**).

If you suspect a problem with these processes, you can look in the *ECE_home***/logs/***Instance_Name*_**GC.log** files for errors, where *Instance_Name* is the name of the ECE process-node instance (the name of the process you defined in the ECE topology file) that you need to troubleshoot. For example, look in **emGateway1_GC.log**. By default, four GC log files are made available in the directory, you can change the number of GC log files available by setting the **numberOfGCLogFiles** in the *ECE_home***/config/ece.properties** file.

You can also disable the generation of GC log files by setting the **enableGCLogs** entry in the *ECE_home***/config/ece.properties** file.

infoCollector Syntax

The infoCollector command syntax is:

```
infoCollector [-v] [-nc] [-l] [-gc][-d dir] [-t]["SUBSCRIPTION|SUBSCRIPTION.SESSION
IDENTIFIER", "..."] [-td][-s] [-e "file_filter", "file_filter2", "..."]
```

where:

- **-v** outputs information pertaining to the collected files.
- **-nc** does *not* compress the resulting directory into a TAR file.
- **-l** includes all log files into the collection. If **-l** is not used, the command only collects those log files that match the node-name with the **.log** suffix that you specify.
- **-gc** collects all the GC log files.
- **-d** *dir* specifies the directory to hold the data.
- **-t** adds all the trace files matching the subscription session identifiers to the collection.
- **-t** *"SUBSCRIPTION|SUBSCRIPTION.SESSION IDENTIFIER", "..."* adds all the trace files matching the subscription session identifiers to the collection.
- **-td** *dir* collects all the trace files from the provided directory or location.
- **-s** includes all files from the *ECE_home***/sample_data** directory.
- **-e "***file_filter***"** is the path name or its pattern of custom directories or files you want to collect and include in the compressed TAR file.

  Separate multiple filters with a comma.

  For example, assume in your *ECE_home* directory you have the files **notes.txt**, **comments_1.txt**, **comments_2.txt**, and **comments_3.txt**, and you also have a directory named **observations** that contains the files **observation1.txt**, **observation2.txt**, and **observation3.txt**.

  The path name pattern can be either an explicit file name such as **notes.txt**, an entire directory tree such as **observations**, or a wildcard ***** such as shown for the files that begin with **comments** in the following example:

  ```
  infoCollector -e "/home/example/ECE_11.2.0.2/notes.txt",
  "/home/example/ECE_11.2.0.2/observations", "/home/example/ECE_11.2.0.2/comments*"
  ```

  The preceding **infoCollector** command would put all custom files and directories into a directory named **extra_files** and the resulting collection of files would have the following directory structure:

```
infoCollector_2014-05-07T11:02:10
  localhost-DRIVER
    extra_files
      observations
        observation3.txt
        observation2.txt
        observation1.txt
      notes.txt
      comments_3.txt
      comments_2.txt
      comments_1.txt
    config
    brm_config
    VERSION
```

Troubleshooting Performance Issues by Using Coherence JMX Metrics

ECE provides Coherence metrics that can help you troubleshoot performance problems and performance tuning, and isolate hardware issues.

To troubleshoot performance issues by using Coherence JMX metrics:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **Coherence** node.
3. Expand **Service**.
4. View the Coherence JMX metrics that apply to your troubleshooting scenario:
   - Expand **InvocationService**, select the appropriate node, and expand **Attributes**.

     The **TaskCount** attribute specifies the number of request batches received by the node.

     The **TaskAverageDuration** attribute specifies the average request batch latency for the node.
   - Expand **BRMDistributedCache**, select the appropriate node, and expand **Attributes**.

     The **RequestTotalCount** attribute specifies the following, depending on the request type:
     - For Initiate, Update, Terminate, and Cancel requests, it specifies the number of entry processor invocations.
     - For Auth Query requests, it specifies the number of get() operations.
   - The **RequestAverageDuration** attribute specifies the following, depending on the request type:
     - For Initiate, Update, and Terminate requests, it specifies entry processor latency.
     - For Auth Query requests, it specifies get latency.

To reset the attribute values, expand **Operations**, select the **resetStatistics** operation, and then click the **resetStatistics** button.

Troubleshooting Failed Usage Requests

ECE may occasionally fail to process usage requests. For example, a usage request could fail because the customer does not own a relevant charge offer. To help you troubleshoot the reason for a failure, ECE, HTTP Gateway, and Diameter Gateway use log4j to collect information about failed usage requests, such as:

- Reason for the failure
- Customer identifier
- Session identifier

For example, log4j could log the following information about a failed usage request in Diameter Gateway:

```
ERROR - - - - Failing Usage Request for subscriber ID : <<PUID>>, session
ID :<<Session ID>,
reasons : [NO_QUALIFIED_CHARGE_OFFERS, ZERO_RUM_QUANTITY]
```

You can use these log files to determine the reason for a failure, so you can fix any issues before reprocessing the usage request.

> **Note:**
>
> Log4j logs usage request errors for ECE, HTTP Gateway, and Diameter Gateway even when debug mode is not enabled.

Troubleshooting Problems with Rating

ECE returns a failure message with a new reason code, NO_RATING_GRAPH_CONFIGURED, if the graph for the RUM configured for rating could not be found in the path specified. However, in multiple RUMs scenario, if the graph could not be found for any of the RUMs configured for rating, ECE uses the NO_RATED_QUANTITY reason code to indicate the rating failure.

By default, if the graph is missing, ECE considers it as an error and returns a failure message with the NO_RATING_GRAPH_CONFIGURED reason code. However, you can configure ECE to not consider this as an error by setting the **treatNoRatingGraphAsError** attribute in the *ECE_home*/**config/management/charging-settings.xml** file to **false**. When set to **false**, ECE reports the NO_RATING_GRAPH_CONFIGURED reason code in its response and continues rating.

To skip missing graph and continue rating:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **charging.server**.
4. Expand **Attributes**.
5. Set the **treatNoRatingGraphAsError** attribute to **false**.

Troubleshooting Problems with Rerating

Rerating errors are handled as follows for each rerating stage:

- **In the prepare to rerate stage**

  Errors are logged in the **CustomerUpdater.log** file. No acknowledgement is sent so the acknowledgement queue is empty.

- **During rerating**

  Errors are logged in the **emGateway.log**.

- **In the rerate complete stage**

  Errors are logged in the **CustomerUpdater.log** file. ECE sends a notification to the BRM server using BRM Gateway to create a new rerate job.

Diameter Gateway Error Codes

This appendix describes Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) Diameter Gateway error codes.

Topics in this document:

- Success Result Codes
- Protocol-Error Result Codes
- Transient-Failure Result Codes
- Permanent-Failure Result Codes

See also "Adding Diameter Gateway Nodes for Online Charging" and "Configuring Diameter Gateway Nodes".

For more information, see the Result-Code AVP section in the IETF website at: `https://tools.ietf.org/html/rfc3588#section-7.1`.

Troubleshooting a Corrupted ECE Configuration File

When you configure ECE charging, subscriber preferences, and so on, ECE stores the information in the *ECE_home*/**config/management/charging-settings.xml** file. This file is loaded into cache when ECE is started, and run-time changes can be made using a JMX editor, such as JConsole.

If the **charging-settings.xml** file is either accidentally deleted or corrupted while the ECE system is up and running, you can rebuild the file from the data available in cache. The rebuilt file will contain the original configuration with any run-time configuration changes.

To rebuild the **charging-settings.xml** file from the data available in ECE cache:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole. See "Accessing ECE Configuration MBeans".
2. Expand the **ECE Configuration** node.
3. Expand **systemAdmin**.

4. Expand **Operations**.
5. Click **rebuildChargingSettingsFile**.

The operation rebuilds the **charging-settings.xml** file and saves it to the path *ECE_home***/config/management/**.

Troubleshooting JVM and Coherence

When troubleshooting ECE, you may need to troubleshoot JVM and Coherence. For troubleshooting JVM issues, note the following:

- See the Java Help Center website:

  https://www.java.com/en/download/help/index.html

- Java VM arguments and properties are defined in the *ECE_home***/config/ece.properties** file.
- Java tuning profiles for nodes are defined in the *ECE_home***/config/defaultTuningProfile.properties** file.
- When ECE is running, you can use JConsole to access JMX information provided by the JVM. See "Using JConsole" in *Java Platform, Standard Edition Monitoring and Management Guide* for more information.

For troubleshooting Coherence issues, note the following:

- You can access runtime information about Coherence through JMX, specifically through the Coherence MBeans (which are used to manage and monitor different parts of Coherence).

  See "Oracle Coherence MBeans Reference" in *Oracle Fusion Middleware Managing Oracle Coherence* for more information.

- See the Oracle Coherence Knowledge Base website:

  https://coherence.java.net/

Troubleshooting Failed Diameter-Message Processing in Diameter Gateway

If you suspect a problem with how Diameter Gateway nodes are processing Diameter messages, look in the *ECE_home***/logs/***instance_name***.log** files for errors, where *instance_name* is the name of the Diameter Gateway-node instance (a name you defined in the ECE topology file) that you need to troubleshoot. For example, look in **diameterGateway1.log**.

To set log levels for Diameter Gateway nodes, obtain the Diameter Gateway module names in the *ECE_home***/config/log4j2.xml** file, and then set the log levels by module as described in "Reading Log Files".

Diameter Gateway returns all Diameter result codes (Result-Codes) as part of the Credit Control Answer (CCA) message. When an error occurs, the error ID and name are returned in the result code. For example, if the CCR was missing an Event-Timestamp AVP, the error would be:

```
DiameterTalk Answers =[
Diameter Message: CCA
```

```
Version: 1
Msg Length: 144
Cmd Flags: PXY
Cmd Code: 272
App-Id: 4
Hop-By-Hop-Id: 1497412149
End-To-End-Id: 734750287
  Session-Id (263,M,l=11) = 111
  Result-Code (268,M,l=12) = DIAMETER_MISSING_AVP (5005)
  Origin-Host (264,M,l=24) = dgw1.example.com
  Origin-Realm (296,M,l=19) = example.com
  Auth-Application-Id (258,M,l=12) = 4
  CC-Request-Type (416,M,l=12) = INITIAL_REQUEST (1)
  CC-Request-Number (415,M,l=12) = 0
  Failed-AVP (279,M,l=20) =
    Event-Timestamp (55,M,l=12) = 3627391363 (Fri Dec 12 08:42:43 PST 2014)
```

For the error, an error message is written to the **diametergateway***Instance_Name***.log** file that indicates the nature and stack trace for the error.

For information about Diameter Gateway result codes, see "Diameter Gateway Error Codes".

Diameter Gateway nodes must be started after the customer data is loaded into the ECE grid; otherwise, they cannot process Diameter requests.

Troubleshooting Failed RADIUS-Message Processing in RADIUS Gateway

If you suspect a problem with how RADIUS Gateway nodes are processing RADIUS messages, see the following files for errors that you need to troubleshoot:

- *ECE_home***logs/***Instance_Name***.log** files, where *Instance_Name* is the name of the RADIUS Gateway-node instance (a name you defined in the ECE topology file); for example, *ECE_home***logs/radiusGateway1.log**.
- Charging-server node log files; for example, *ECE_home***logs/ecs1.log**.

To set log levels for RADIUS Gateway nodes, obtain the RADIUS Gateway module names in the *ECE_home***config/log4j2.xml** file, and then set the log levels by module as described in "Reading Log Files".

RADIUS Gateway returns all the results as part of the reply-message attribute-value pair (AVP) in the RADIUS response. For example, if the user password in the authentication request is incorrect, the following error message is returned in the RADIUS response:

```
Session_Timeout AVP after Deletion : null2016-03-07 23:37:58.896 PST DEBUG -  -
-  - ECE Radius server - Sending the response to client
 Code: Access-Reject(3)
 Identifier: 0
 Length: 20
 Authenticator: 0x00000000000000000000000000000000
 Reply-Message: RadiusGatewayMessagesBundle-31015: Incorrect password from User
 User-Name: 0049100033
```

77

ECE Utilities

Learn about the syntax and parameters for the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) utilities.

Topics in this document:

configLoader

Use the **configLoader** utility to load mediation specification files into ECE. These files are used by Diameter Gateway and RADIUS Gateway. See *ECE Implementing Charging* and "Configuring the configLoader Utility".

Location

*ECE_home***/bin**

Syntax

```
[start] [stop] configLoader
```

Parameters

-help
Displays the syntax for this utility.

Results

Results are written to the *ECE_home***/logs/configLoader.log** file.

customerLoader

On a development system, use the **customerLoader** utility to load customer data into ECE from XML files. See *ECE Implementing Charging* and "Configuring the customerLoader utility".

In a production environment, use the **customerLoader** utility with the **-incremental** parameter to load customer data incrementally.

> **Caution:**
>
> Do *not* run the **customerLoader** utility without the **-incremental** parameter in a production environment.

Location

*ECE_home***/bin**

Syntax

```
{start|stop} customerLoader [-incremental [customerupdaterfilename]] [-help]
```

Parameters

start
Starts **customerLoader**.

stop
Stops **customerLoader**.

-incremental *customerupdaterfilename*
Incrementally loads customer data. Use this parameter to selectively synchronize the BRM database and the ECE cache. This is typically done to manage data migration errors.

-help
Displays the syntax for **customerLoader**.

Results

Results are written to the *ECE_home***/logs/customerLoader.log** file

To verify that the customer objects were loaded in the ECE cluster distributed cache, use the query tool to form CohQL queries on the cache name that stores the respective object. For example, you would look in the Customer cache.

encrypt.sh

Use the **encrypt.sh** utility to encrypt passwords that ECE uses for connecting to BRM and Pricing Design Center (PDC).

For more information, see "About Managing External Application Passwords".

Location

*ECE_home***/bin**

Syntax

```
encrypt.sh string storepassword [-help]
```

Parameters

string
Specifies the new password to encrypt.

storepassword
Specifies the password to the keystore.

-help
Displays the syntax for this utility.

Results

The utility returns a message containing the encrypted value for the new password.

Results are written to the *ECE_home***/logs/encrypt.log** file.

events_propagate_utility.pl

Use the **events_propagate_utility.pl** utility to propagate failed customer data updates from the suspense queue to the BRM Oracle DM database queue.

For more information, see "Managing Failed Customer Data Updates".

Location

*ECE_home***/brm_utils/**

Copy **event_propagate.sql** and **events_propagate_utility.pl** from the ECE installation directory (*ECE_home***/brm_utils/**) to your BRM machine (for example, *BRM_home***/sys/test/**).

Syntax Overview

The following actions are supported for **events_propagate_utility**:

- Syntax for Moving Events from Suspense Queue
- Syntax for Listing All Queues
- Syntax for Loading Data into Database
- Syntax for Purging Data from the Queue
- Syntax for Getting Help

Syntax for Moving Events from Suspense Queue

Moves events from the suspense queue to the BRM Oracle DM database queue.

```
perl events_propagate_utility.pl move -s queue_name -d queue_name [-l user_id/
password@oracle_sid]
                                      -n number_of_events
```

Parameters for Moving Events from the Suspense Queue

The following parameters are used to move from the suspense queue to the BRM Oracle DM database queue:

-s *queue_name*
Specifies the suspense queue name.

-d *queue_name*
Specifies the BRM Oracle DM database queue name.

-l *user_id/password@oracle_sid*
Specifies the user ID and password to connect to the database.

-n *number_of_events*
Specifies the number of events that are being moved from the suspense queue.

Syntax for Listing All Queues

Lists all queues on a BRM machine.

```
perl events_propagate_utility.pl list [-l user_id/password@oracle_sid]
```

Parameters for Listing All Queues

The following parameters are used to list all queues on a BRM machine:

-l *user_id/password@oracle_sid*
Specifies the user ID and password to connect to the database.

Syntax for Loading Data into Database

Loads data from the queue into the database.

```
perl events_propagate_utility.pl load [-l user_id/password@oracle_sid]
```

Parameters for Loading Data into Database

The following parameters are used to load data into the database:

-l *user_id/password@oracle_sid*
Specifies the user ID and password to connect to the database.

Syntax for Purging Data from the Queue

Purges data from the suspense queue or BRM Oracle DM database queue.

```
perl events_propagate_utility.pl purge [-l user_id/password@oracle_sid] -q
queue_name
```

Parameters for Purging Data from Queues Events

The following parameters are used to purge data from the suspense queue and the BRM Oracle DM database queue:

-l *user_id/password@oracle_sid*
Specifies the user ID and password to connect to the database.

-q *queue_name*
Specifies the queue name on which data is purged.

Syntax for Getting Help

Displays the syntax for the **events_propagate_utility** script.

```
perl events_propagate_utility.pl help
```

Results

The **events_propagate_utility** script notifies you when it runs successfully. Otherwise, look in the logs.

gridSync

Use the **gridSync** utility in a disaster-recovery system. You use the **gridSync** utility to start the federation service to replicate data to the participant sites asynchronously and to also replicate all existing ECE cache data to the participant sites. See "Configuring ECE for Disaster Recovery".

Location

*ECE_home***/bin**

Syntax

```
[-help] gridSync {start|stop|pause|replicate} [remoteClusterName]
```

Parameters

-help
Displays the syntax.

start
Starts the federation service to replicate the ECE cache data to the backup or remote production sites.

stop
Stops replicating ECE cache data to the backup or remote production sites.

pause
Pauses replication of ECE cache data to the backup or remote production sites.

replicate
Replicates all the existing ECE cache data to the backup or remote production sites.

remoteClusterName
Specify the name of the Coherence cluster in the backup site.

Results

Results are written to the *ECE_home***/logs/encrypt.log** file.

pricingLoader

Use the **pricingLoader** utility to load sample pricing data into ECE from an XML file. On a development system, you use **pricingLoader** after ECE starts up to prime its cache with sample pricing data.

See *ECE Implementing Charging* and "Configuring the pricingLoader Utility".

> **Note:**
>
> - Do *not* mix running **pricingLoader** and Pricing Updater on an ECE deployment.
> - Use **pricingLoader** only on a development system.

Location

ECE_home/**bin**

Syntax

```
[start] [stop] pricingLoader
```

Parameters

-help
Displays the syntax.

A

WebLogic-Based Application Metrics

This appendix describes how WebLogic Monitoring Exporter collects metrics in the following groups to produce data for monitoring PDC, Business Operations Center, Billing Care, and Billing Care REST API in Prometheus and Grafana:

- WLS Server Metrics Group
- Application Runtime Metric Group
- Servlets Metric Group
- JVM Runtime Metric Group
- Execute Queue Runtimes Metric Group
- Work Manager Runtimes Metric Group
- Thread Pool Runtime Metric Group
- JDBC Service Runtime Metric Group
- JTA Runtime Metric Group
- WLS Scrape MBean Metric Group
- Persistent Store Runtime MBean Metric Group

WLS Server Metrics Group

Use the WLS server metrics group to retrieve runtime information about a server instance and to transition a server from one state to another. Table A-1 lists the metrics in this group.

Table A-1 WLS Server Metrics

Metric Name	Label	Metric Type	Description
wls_server_activation_time	location	long	Returns the time when the server was started.
wls_server_admin_server_listen_port	location	int	Returns the port on which this server is listening for requests.
wls_server_open_sockets_current_count	location	int	Returns the current number of sockets registered for socket muxing on this server.
wls_server_state_val	location	int	Returns the current state of the server as an integer: • **0**: Shutdown • **1**: Starting • **2**: Running

Application Runtime Metric Group

Use the application runtime metric group to collect runtime information about a deployed enterprise application. Table A-2 describes the metrics in the group.

Table A-2 Application Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_webapp_config_deployment_state	location app name	int	Returns the current state of the deployment as an integer.
wls_webapp_config_open_sessions_current_count	location app name	int	Returns the current number of open sessions in this module.
wls_webapp_config_open_sessions_high_count	location app name	int	Returns the highest number of open sessions on this server at any one time.
wls_webapp_config_sessions_opened_total_count	location app name	int	Returns the total number of sessions that were opened.

Servlets Metric Group

Each WAR file can contain multiple servlets, and each WAR file can be integrated into an enterprise archive (EAR). Use the servlets metric group to obtain runtime information about a web application and each servlet. Table A-3 describes the metrics in this group.

Table A-3 Servlets Metrics

Metric Name	Label	Metric Type	Description
wls_servlet_execution_time_average	location app name servlet Name	long	Displays the average amount of time, in milliseconds, it took to execute all invocations of the servlet since it was most recently deployed.
wls_servlet_execution_time_high	location app name servlet Name	long	Displays the average amount of time, in milliseconds, that the single longest invocation of the servlet has executed since it was most recently deployed.

Table A-3 (Cont.) Servlets Metrics

Metric Name	Label	Metric Type	Description
wls_servlet_execution_time_low	location app name servlet Name	long	Displays the average amount of time, in milliseconds, that the single shortest invocation of the servlet has executed since it was most recently deployed.
wls_servlet_execution_time_total	location app name servlet Name	long	Displays the average amount of time, in milliseconds, that all invocations of the servlet have executed since it was most recently deployed.
wls_servlet_invocation_total_count	location app name servlet Name	int	Displays the total number of times the servlet has been invoked since WebLogic Server started.
wls_servlet_pool_max_capacity	location app name servlet Name	int	Displays the maximum capacity of this servlet for single thread model servlets.
wls_servlet_reload_total_count	location app name servlet Name	int	Displays the total number of times WebLogic Server has reloaded the servlet since it was last deployed. WebLogic Server typically reloads a servlet if it has been modified.

JVM Runtime Metric Group

Use the JVM runtime metric group to retrieve information about the Java Virtual Machine (JVM) that the current server instance is running. Table A-4 describes the metrics in this group.

Table A-4 JVM Runtime Metrics

Metric Name	Labels	Metric Type	Description
wls_jvm_heap_free_current	name	long	Returns the current amount of memory, in bytes, that is available in the JVM heap.
wls_jvm_heap_free_percent	name	int	Returns the percentage of the JVM heap that is free.
wls_jvm_heap_size_current	name	long	Returns the current size, in bytes, of the JVM heap.
wls_jvm_heap_size_max	name	long	Returns the maximum size, in bytes, of the JVM heap.

Table A-4 (Cont.) JVM Runtime Metrics

Metric Name	Labels	Metric Type	Description
wls_jvm_process_cpu_load	name	time	Returns the amount of CPU time that the Java virtual machine is running in nanoseconds.
wls_jvm_uptime	name	long	Returns the number of milliseconds that the virtual machine has been running.

Execute Queue Runtimes Metric Group

Use the execute queue runtime metric group to return information about the queue. Table A-5 describes the metrics in this group.

Table A-5 Execute Queue Runtimes Metrics

Metric Name	Labels	Metric Type	Description
wls_socketmuxer_pending_request_current_count	name	int	Returns the number of waiting requests in the queue.

Work Manager Runtimes Metric Group

Use the work manager runtimes metric group to retrieve information about requests from the work manager. Table A-6 describes the metrics in this group.

Table A-6 Work Manager Runtimes Metrics

Metric Name	Label	Metric Type	Description
wls_workmanager_completed_requests	name	int	Returns the number of requests that have been processed.
wls_workmanager_pending_requests	name	int	Returns the number of waiting requests in the queue.
wls_workmanager_stuck_thread_count	name	int	Returns the number of stuck threads in the thread pool.

Thread Pool Runtime Metric Group

Use the thread pool runtime metric group to monitor the self-tuning queue. Table A-7 describes the metrics in this group.

Table A-7 Thread Pool Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_threadpool_execute_thread_total_count	name	int	Returns the total number of threads in the pool.
wls_threadpool_hogging_thread_count	name	int	Returns the threads that are currently being held by a request. These threads will either be declared as stuck after the configured timeout period or be returned to the pool.
wls_threadpool_queue_length	name	int	Returns the number of pending requests in the priority queue.
wls_threadpool_stuck_thread_count	name	int	Returns the number of stuck threads in the thread pool.

JDBC Service Runtime Metric Group

Use the JDBC service runtime metric group to retrieve runtime information about a server instance and to transition a server from one state to another. Table A-8 describes the metrics in this group.

Table A-8 JDBC Service Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_datasource_active_connections_average_count	name	int	Returns the average number of active connections in this data source instance.
wls_datasource_active_connections_current_count	name	int	Returns the number of connections currently in use by applications.
wls_datasource_active_connections_high_count	name	int	Returns the highest number of active database connections in this data source instance since the data source was instantiated.
wls_datasource_commit_outcome_retry_total_count	name	int	Returns the cumulative total number of commit outcome query retries conducted before resolving the outcome or exceeding the retry seconds in this data source since the data source was deployed.
wls_datasource_connection_delay_time	name	int	Returns the average amount of time, in milliseconds, that it takes to create a physical connection to the database.
wls_datasource_connections_total_count	name	int	Returns the cumulative total number of database connections created in this data source since the data source was deployed.
wls_datasource_curr_capacity_high_count	name	int	Returns the highest number of database connections available or in use (current capacity) in this data source instance since the data source was deployed.
wls_datasource_curr_capacity	name	int	Returns the current count of JDBC connections in the data source's connection pool.

Table A-8 (Cont.) JDBC Service Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_datasource_deployment_state	name	int	Returns the module's current deployment state.
wls_datasource_failed_repurpose_count	name	int	Returns the number of repurpose errors that have occurred since the data source was deployed.
wls_datasource_failed_reserve_request_count	name	int	Returns the cumulative running count of connection requests from this data source that could not be fulfilled.
wls_datasource_failures_to_reconnect_count	name	int	Returns the number of times that the data source attempted to refresh a database connection and failed.
wls_datasource_highest_num_available	name	int	Returns the highest number of database connections that were idle and available to be used by an application at any time in this data source instance since the data source was deployed.
wls_datasource_highest_num_unavailable	name	int	Returns the highest number of database connections that were in use by applications or being tested by the system in this data source instance since the data source was deployed.
wls_datasource_leaked_connection_count	name	int	Returns the number of leaked connections.
wls_datasource_num_available	name	int	Returns the number of database connections that are currently idle and available to be used by applications in this data source instance.
wls_datasource_num_unavailable	name	int	Returns the number of connections currently in use by applications or being tested in this data source instance.
wls_datasource_prep_stmt_cache_access_count	name	long	Returns the cumulative, running count of the number of times that the statement cache was accessed.
wls_datasource_prep_stmt_cache_add_count	name	long	Returns the cumulative, running count of the number of statements added to the statement cache.
wls_datasource_prep_stmt_cache_current_size	name	int	Returns the number of prepared and callable statements currently cached in the statement cache.
wls_datasource_prep_stmt_cache_delete_count	name	long	Returns the cumulative, running count of statements discarded from the cache.
wls_datasource_prep_stmt_cache_hit_count	name	long	Returns the cumulative, running count of the number of times that statements from the cache were used.
wls_datasource_prep_stmt_cache_miss_count	name	long	Returns the number of times that a statement request could not be satisfied with a statement from the cache.
wls_datasource_reserve_request_count	name	long	Returns the cumulative, running count of connection requests from this data source.

Table A-8 (Cont.) JDBC Service Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_datasource_waiting_for_connection_current_count	name	int	Returns the number of connection requests waiting for a database connection.
wls_datasource_waiting_for_connection_failure_total	name	long	Returns the cumulative, running count of connection requests from this data source that had to wait before getting a connection and eventually failed to get a connection.
wls_datasource_waiting_for_connection_high_count	name	int	Returns the highest number of application requests concurrently waiting for a connection from this data source instance.
wls_datasource_waiting_for_connection_success_total	name	long	Returns the cumulative, running count of connection requests from this data source that had to wait before getting a successful connection.
wls_datasource_waiting_for_connection_total	name	long	Returns the cumulative, running count of connection requests from this data source that had to wait before getting a connection. This includes requests that eventually got a connection and those that did not get a connection.

JTA Runtime Metric Group

Use the JTA runtime metric group to access transaction runtime characteristics within a WebLogic Server. Table A-9 describes the metrics in this group.

Table A-9 JTA Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_jta_active_transactions_total_count	name	long	Returns the number of active transactions on the server.
wls_jta_seconds_active_total_count	name	int	Returns the total number of seconds that transactions were active for all committed transactions.
wls_jta_transaction_abandoned_total_count	name	long	Returns the total number of transactions that were abandoned since the server was started.
wls_jta_transaction_committed_total_count	name	long	Returns the total number of transactions committed since the server was started.
wls_jta_transaction_heuristics_total_count	name	long	Returns the number of transactions that completed with a heuristic status since the server was started.
wls_jta_transaction_llrcommitted_total_count	name	long	Returns the total number of LLR transactions that were committed since the server was started.
wls_jta_transaction_no_resources_committed_total_count	name	long	Returns the total number of transactions with no enlisted resources that were committed since the server was started.

Table A-9 (Cont.) JTA Runtime Metrics

Metric Name	Label	Metric Type	Description
wls_jta_transaction_one_resource_one_phase_committed_total_count	name	long	Returns the total number of transactions with more than one enlisted resource that were one-phase committed due to read-only optimization since the server was started.
wls_jta_transaction_total_count	name	long	Returns the total number of transactions processed. This total includes all committed, rolled back, and heuristic transaction completions since the server was started.

WLS Scrape MBean Metric Group

Use the WLS scrape metric group to monitor the performance of the WebLogic Server. Table A-10 describes the metrics in this group.

Table A-10 WLS Scrape MBean Metrics

Metric Name	Label	Metric Type	Description
wls_scrape_mbeans_count_total	instance	long	Returns the number of metrics scraped.
wls_scrape_duration_seconds	instance	long	Returns the time required to do the scrape.
wls_scrape_cpu_seconds	instance	long	Returns the amount of time the CPU used during the scrape.

Persistent Store Runtime MBean Metric Group

Use the persistent store runtime MBean metric group to monitor a persistent store. Table A-11 describes the metrics in this group.

Table A-11 Persistent Store Runtime MBean Metrics

Metric Name	Label	Metric Type	Description
wls_persistentstore_allocated_io_buffer_bytes	name	long	Returns the amount of off-heap (native) memory, in bytes, reserved for file store use. When applicable, this is a multiple of the file store configurable attribute IOBufferSize. This applies to synchronous write policies Direct-Write and Cache-Flush policies.

Table A-11 (Cont.) Persistent Store Runtime MBean Metrics

Metric Name	Label	Metric Type	Description
wls_persistentstore_allocated_window_buffer_bytes	name	long	Returns the amount of off-heap (native) memory, in bytes, reserved for file store window buffer use. Applies to synchronous write policies Direct-Write-With-Cache and Disabled, but only when the native wlfileio library is loaded.
wls_persistentstore_create_count	name	long	Returns the number of create requests issued by this store.
wls_persistentstore_delete_count	name	long	Returns the number of delete requests issued by this store.
wls_persistentstore_object_count	name	int	Returns the number of objects contained in the connection.
wls_persistentstore_physical_write_count	name	long	Returns the number of times the store flushed its data to durable storage.
wls_persistentstore_read_count	name	long	Returns the number of read requests issued by this store, including requests that occur during store initialization.
wls_persistentstore_update_count	name	long	Returns the number of update requests issued by this store.

B

ECE Directory Structure and Contents

This appendix describes the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) directory structure and directory contents.

Topics in this document:

ECE Server Software

Table B-1 shows the ECE server software directory structure where *ECE_home* is the directory in which ECE is installed.

Table B-1 Elastic Charging Engine Directory Structure and Contents

Directory	Description
ECE_home/**bin**	Contains scripts. Path to this directory to run Elastic Charging Controller (ECC).
ECE_home/**config**	Contains security-related configuration files.
ECE_home/**config/management**	Configuration files. **Important:** All the configurations defined in these files should be viewed and changed through JMX.
ECE_home/**lib**	Libraries such as JAR files and binaries.
ECE_home/**logs**	Log files.
ECE_home/**sample_data**	Sample data used by ECE for a development system.
ECE_home/**sample_data/config_data**	Sample data used by ECE for a development system.
ECE_home/**sample_data/config_data/specifications/ece_end2end**	Sample ECE event definition data for a development system.
ECE_home/**sample_data/crossref_data**	Sample data used by ECE for a development system.
ECE_home/**sample_data/customer_data**	Sample data used by ECE for a development system.
ECE_home/**sample_data/policy_data**	Sample data that can be used for testing the integration of policy clients with ECE.
ECE_home/**sample_data/pricing_data**	Sample data used by ECE for a development system.

ECE SDK

Table B-2 shows the ECE SDK software directory structure where *ECE_home* is the directory in which the ECE Server software is installed.

Table B-2 Elastic Charging Engine SDK Directory Structure and Contents

Directory	Description
ECE_home/**ocecesdk/bin**	Directories that contain shell scripts for compiling and running various types of sample programs.
ECE_home/**ocecesdk/bin/extensions**	Shell scripts for extension-implementation sample programs.
ECE_home/**ocecesdk/bin/notification**	Shell scripts for notification sample programs.
ECE_home/**ocecesdk/bin/policy**	Shell scripts for policy sample programs.
ECE_home/**ocecesdk/bin/query**	Shell scripts for query sample programs.
ECE_home/**ocecesdk/bin/update**	Shell scripts for update sample programs.
ECE_home/**ocecesdk/bin/usage**	Shell scripts for usage sample programs.
ECE_home/**ocecesdk/config**	Configuration files common to all sample programs.
ECE_home/**ocecesdk/config/extensions**	Configuration files for extension-implementation sample programs.
ECE_home/**ocecesdk/source** **Note:** The full path showing the Java project directory structure to the sample programs is: *ECE_home*/**ocecesdk/source/oracle/communication/brm/charging/sdk**	Java sample programs.
ECE_home/**ocecesdk/source/.../sdk/extensions**	Source files for extensions sample programs. Includes the data loader used for extensions.
ECE_home/**ocecesdk/source/.../sdk/notification**	Source files for notification sample programs.
ECE_home/**ocecesdk/source/.../sdk/policy**	Source files for policy sample programs.
ECE_home/**ocecesdk/source/.../sdk/query**	Source files for query sample programs.
ECE_home/**ocecesdk/source/.../sdk/update**	Source files for update sample programs.
ECE_home/**ocecesdk/source/.../sdk/usage**	Source files for usage sample programs.

C

ECC Commands

This appendix describes the Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) Elastic Charging Controller (ECC) commands.

Topics in this document:

- ECC Commands

See also "ECE System Administration Overview ".

ECC Commands

Table C-1 shows the ECC commands and their descriptions.

Table C-1 Elastic Charging Controller (ECC) Commands

Command	Description
addNode	Add an ECE node. addNode does not start the node. You must use the start command to start the node. See "Adding Nodes".
cat	Concatenate and print files. This is a standard shell command.
encrypt	Encrypt the given string. See "About Managing External Application Passwords".
:exit	Exit the shell. This is a standard shell command.
:help	Get help for commands. This is a standard shell command.
:history	Get history. This is a standard shell command.
infoCollector	Collect log files. See "Collecting Log Files for Sending to Oracle Technical Support".
kill	Same as bash kill. This is a standard shell command.
ls	Same as bash ls. This is a standard shell command.
removeNode	Remove an ECE node. You must stop the node before running removeNode. See "Removing Nodes".
rollingUpgrade	Perform a rolling upgrade of ECE. See *BRM Installation Guide*.

Table C-1 (Cont.) Elastic Charging Controller (ECC) Commands

Command	Description
status	Displays if nodes are started or stopped. See "Checking If Nodes are Started or Stopped".
start	Start nodes by name or by role. See "Starting and Stopping ECE ".
stop	Stop nodes by name or by role. See "Starting and Stopping ECE ".
sync	Install ECE on all remote hosts as specified in the topology file. See *BRM Installation Guide*.
tail	Display the last part of a file. This is a standard shell command.

D

ECE Configuration File Reference

This appendix describes Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) configuration files.

Topics in this document:

System Configuration Files

Table D-1 summarizes configuration files relevant to tuning the ECE system, such as tuning Oracle Coherence and JVM parameters. All system configuration files are in *ECE_home*/**config**.

Table D-1 ECE System Configuration File Reference

File Name	Description
charging-cache-config.xml	Used for configuring Coherence.
charging-coherence-override-dev.xml	Used for configuring Coherence. Used for development.
charging-coherence-override-prod.xml	Used for configuring Coherence. Used for production.
charging-coherence-override-secure-prod.xml	Used for configuring Coherence. Used for production. When you enable SSL, the key and store passwords must be set in this file.
charging-context.xml	System configuration file. Do not modify.
charging-pof-config.xml	System configuration file. Do not modify.
coherence-jaas.config	Used for Java security. Do not modify.
defaultTuningProfile.properties	Used for setting JVM tuning parameters, such as those for garbage collection and heap size. Typically, the default values do not need to be modified. See "Configuring JVM Tuning Parameters".
ece.properties	System configuration file.
eceTopology.conf	Used for specifying the physical server machines for each Coherence node in the cluster.
groovysh.profile	System configuration file. Do not modify.

Table D-1 (Cont.) ECE System Configuration File Reference

File Name	Description
JMSConfiguration.xml	Used for configuring the JMS credentials for the JMS server on which the notification queue (JMS topic) resides. See "Modifying JMS Credentials for Publishing External Notifications" in *ECE Implementing Charging*.
jmxremote.password	Used for JXM security. See "Performing a Secure ECE Installation " in *BRM Security Guide*.
jrds.groovy	System configuration file. Do not modify.
log4j2.xml	Used for configuring logging for each node in the cluster. This file is used only initially before the ECE system is running. After ECE is running, you use JConsole to set logging.
Notification.xsd	XSD schema file for XML notification messages generated by ECE.
permissions.xml	Used for authorization of nodes in the Coherence cluster. See "Performing a Secure ECE Installation " in *BRM Security Guide*.
sdkTuningProfile.properties	Used for configuring tuning parameters for the sample programs included in the ECE SDK. Typically, the default values do not need to be modified.
server.jks	Used for authentication of nodes in the Coherence cluster (Coherence cluster security). Do not modify. See "Setting Up and Managing Elastic Charging Engine Security ".

Business Configuration Files

Table D-2 summarizes configuration files relevant to setting ECE charging business parameters and runtime configurations. All business configuration files are in *ECE_home***/config/management**.

For information about configuring usage-charging parameters, see "Managing Online Charging Sessions" in *BRM ECE Implementing Charging*.

Table D-2 ECE Business Configuration File Reference

File Name	Description
charging-settings.xml	Used for configuring charging business rules and runtime configurations. See "Managing Online Charging Sessions" in *BRM ECE Implementing Charging*.
migration-configuration.xml	Used for specifying the directories from which you want the data-loading utilities (configuration loader, pricing loader, and customer loader) to load data.

Table D-2 (Cont.) ECE Business Configuration File Reference

File Name	Description
test-tools.xml	Used for specifying developer tool settings.

E

Diameter Gateway Error Codes

This appendix describes Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE) Diameter Gateway error codes.

Topics in this document:

See also "Adding Diameter Gateway Nodes for Online Charging" and "Configuring Diameter Gateway Nodes".

For more information, see the Result-Code AVP section in the IETF website at: `https://tools.ietf.org/html/rfc3588#section-7.1`.

Success Result Codes

Success result codes inform the peer that a request has been successfully completed.

Table E-1 lists the success result codes.

Table E-1 Success Result Codes

Result Code	Code	Notes
DIAMETER_SUCCESS	2001	The requests was successfully completed.
DIAMETER_LIMITED_SUCCESS	2002	The request was successfully completed, but additional processing is required by the application to provide service to the user.

Protocol-Error Result Codes

Protocol-error result codes are returned at the base protocol level. The diameter proxy agents may attempt to correct these errors.

Table E-2 lists the protocol-error result codes.

Table E-2 Protocol-Error Result Codes

Result Code	Code	Notes
DIAMETER_COMMAND_UNSUPPORTED	3001	The command-code in the diameter request is not recognized or supported by the diameter node. Verify that the command-code sent by the peer conforms to Gy, Sh, or Sy protocol specifications.

Table E-2 (Cont.) Protocol-Error Result Codes

Result Code	Code	Notes
DIAMETER_UNABLE_TO_DELIVER	3002	The origin and destination information in the diameter message header is incorrect. Verify that the targeted diameter peer node is run with the correct realm and Origin-Host names.
DIAMETER_REALM_NOT_SERVED	3003	The destination realm in the diameter message header is not recognized. Verify that the message realm name matches the realm configuration in Diameter Gateway.
DIAMETER_TOO_BUSY	3004	The request cannot be processed within the time-out period of Elastic Charging Server. The charging server nodes may be overloaded. Check the CPU usage on the charging server nodes.
DIAMETER_LOOP_DETECTED	3005	An agent detected a loop while trying to get the message to the intended recipient. The message may be sent to an alternate peer, if one is available, but the peer reporting the error has identified a configuration problem.
DIAMETER_REDIRECT_INDICATION	3006	A redirect agent has determined that the request could not be satisfied locally and the initiator of the request should direct the request directly to the server, whose contact information has been added to the response. When set, the Redirect-Host AVP MUST be present.
DIAMETER_APPLICATION_UNSUPPORTED	3007	The Diameter Gateway server received the diameter request with the Auth-Application-Id other than Gy, Sy, or Sh. Verify that the Auth-Application-Id in the diameter request is either Gy (4), Sy (16777302), or Sh attribute-value pair (16777217).
DIAMETER_INVALID_HDR_BITS	3008	The diameter message header contains bits of unexpected byte size, or it is corrupted. Verify the diameter message header.
DIAMETER_INVALID_AVP_BITS	3009	The diameter message contains the attribute-value pair (AVP) bits of unexpected byte size, or it is corrupted. Verify the diameter message AVP stream.
DIAMETER_UNKNOWN_PEER	3010	A CER message was received from an unknown peer.

Transient-Failure Result Codes

Transient-failure result codes are returned to indicate that the request could not be processed at the time it was received. These requests may be processed in the future.

Table E-3 lists the transient-failure result codes.

Table E-3 Transient-Failure Result Codes

Result Code	Code	Notes
DIAMETER_AUTHENTICATION_REJECTED	4001	The authentication process for the user failed, most likely due to an invalid password used by the user. Further attempts MUST only be tried after prompting the user for a new password.
DIAMETER_OUT_OF_SPACE	4002	A Diameter node received the accounting request but was unable to commit it to stable storage due to a temporary lack of space.
ELECTION_LOST	4003	The peer lost the election process and has therefore disconnected the transport connection.
DIAMETER_END_USER_SERVICE_DENIED	4010	The rating function failed with one of the following errors: • MISSING_SYSTEM_ALTERATION_FOR_DEBIT • INCORRECT_IMPACTS_FOR_DEBIT & LIFECYCLE_VALIDATION_FAILED Check the **ecs.log** file for the rating function failure details.
DIAMETER_CREDIT_LIMIT_REACHED	4012	The subscriber's credit floor or ceiling limit is breached. Update the subscriber's credit.
SPLIT_BRAIN_SUSPECTED	4999	This is a Camiant-specific result code which should be returned server when a primary connection already exists and secondary connection is being attempted.

Permanent-Failure Result Codes

Permanent-failure result codes are returned to indicate that the requests will not be processed and that further requests from the diameter client will not be accepted for processing.

Table E-4 lists the permanent-failure result codes.

Table E-4 Permanent-Failure Result Codes

Result Code	Code	Notes
DIAMETER_AVP_UNSUPPORTED	5001	The Diameter Gateway server received a message with an unknown AVP. Verify that the AVP is present in the **dictionary_main.xml** file.
DIAMETER_UNKNOWN_SESSION_ID	5002	The Sy request contains an unknown Session-Id. Verify that the Session-Id is the same as in the initial request.
DIAMETER_AUTHORIZATION_REJECTED	5003	The Diameter Gateway authorization failed with the data in the request message AVP. Check the log files for more information about the authorization failure.

Table E-4 (Cont.) Permanent-Failure Result Codes

Result Code	Code	Notes
DIAMETER_INVALID_AVP_VALUE	5004	The request AVP data cannot be converted to an expected PayloadItem type. Verify that the AVP type in the request is compatible with the PayloadItem type in ECE.
DIAMETER_MISSING_AVP	5005	The Diameter Gateway server received a request without the Gy, Sy, or Sh AVP. Verify that the request contains the required AVPs as per the 3GPP specifications.
DIAMETER_RESOURCES_EXCEEDED	5006	The request cannot be authorized because the user has expended all allowed resources. For example, a user restricted to one dial-up PPP port attempted to establish a second PPP connection.
DIAMETER_CONTRADICTING_AVPS	5007	The Home Diameter server detected AVPs in the request that contradicted each other, and is not willing to provide service to the user. One or more Failed-AVP AVPs MUST be present, containing the AVPs that contradicted each other.
DIAMETER_AVP_NOT_ALLOWED	5008	A diameter message was received with an AVP that is not allowed as per the **dictionary_main.xml** file. Check for any AVP cardinality rules violations as defined in the **dictionary_main.xml** file.
DIAMETER_AVP_OCCURS_TOO_MANY_TIMES	5009	The AVP-to-ECE PayloadItem mapping violated the cardinality mapping rules per the event definition. Check for cardinality rule violations in the AVP-to-PayloadItem mapping.
DIAMETER_NO_COMMON_APPLICATION	5010	The Diameter Gateway server received a Capabilities-Exchange-Request (CER) message requesting applications other than Gy, Sy, or Sh. Check the Diameter Gateway server connection.
DIAMETER_UNSUPPORTED_VERSION	5011	The Diameter Gateway server received a request message with a version number that is not supported. Verify that the Diameter client version is compatible with the Diameter Gateway server version.
DIAMETER_UNABLE_TO_COMPLY	5012	An unexpected system error occurred on the Online Charging System (OCS). Check the ECE and Diameter Gateway log files for details about the error.
DIAMETER_INVALID_BIT_IN_HEADER	5013	This error is returned when an unrecognized bit in the Diameter header is set to one (1).
DIAMETER_INVALID_AVP_LENGTH	5014	The AVP in the request message contains more or fewer bytes than expected by the Diameter Gateway server. Verify that the AVP data types are configured with the correct byte size.

Table E-4 (Cont.) Permanent-Failure Result Codes

Result Code	Code	Notes
DIAMETER_INVALID_MESSAGE_LENGTH	5015	This error is returned when a request is received with an invalid message length.
DIAMETER_INVALID_AVP_BIT_COMBO	5016	The request contained an AVP with which is not allowed to have the given value in the AVP Flags field. A Diameter message indicating this error MUST include the offending AVPs within a Failed-AVP AVP.
DIAMETER_NO_COMMON_SECURITY	5017	A CER message was processed without security mechanism. Verify that security is enabled in Diameter Gateway Communications Layer.
DIAMETER_USER_UNKNOWN	5030	ECE was not able to locate the subscriber ID in the diameter request. Verify that the subscriber exists in ECE.
DIAMETER_RATING_FAILED	5031	The rating function failed because it received one of the following messages: • INSUFFICIENT_RATED_QUANTITY • NO_QUALIFIED_CHARGE_OFFERS • NO_RATED_QUANTITY • ZERO_RUM_QUANTITY Check the **esc.log** file for the rating function failure.
DIAMETER_ERROR_UNKNOWN_POLICY_COUNTERS	5570	ECE does not recognize one or more policy counters specified in the diameter request. Verify that the ECE server is configured for tracking the policy counters requested by the diameter message.
DIAMETER_ERROR_SUBS_DATA_ABSENT	5106	ECE received a request for some profile data that is unknown. Verify that ECE is configured to return the corresponding profile data for the Diameter request.