

**Oracle® Communications
Billing and Revenue Management**

System Administrator's Guide

Release 7.5

E16719-25

December 2019

Copyright © 2011, 2019, Oracle and/or its affiliates. All rights reserved.

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

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

Contents

Preface	xxxi
Audience	xxxi
Accessing Oracle Communications Documentation	xxxi
Documentation Accessibility	xxxi
Document Revision History	xxxi

Part I Basic BRM System Administration

1 Starting and Stopping the BRM System

About Starting and Stopping BRM Components	1-1
Choosing the User Name for BRM Components	1-1
Starting Several BRM Components in Sequence	1-1
Stopping Several BRM Components in Sequence	1-3
Starting a Component by Using the pin_ctl Utility	1-3
Starting BRM Components with pin_ctl	1-4
Starting Optional BRM Components with pin_ctl	1-4
Starting Pipeline Manager Components with pin_ctl	1-4
Getting Diagnostic Data When You Start a Component	1-4
Halting and Restarting a Component with One Command	1-5
Starting a Component and Clearing the Log File	1-5
Starting a Base Set of BRM Components	1-5
Starting BRM Components Automatically	1-6
Starting Multiple Families of BRM Components on the Same Computer	1-6
Confirming That a BRM Component Started Successfully	1-7
Stopping a BRM Component by Using the pin_ctl Utility	1-7
Getting Diagnostic Data When You Stop a Component	1-7
Stopping a Process by Using Commands	1-8
Stopping a Base Set of BRM Components	1-8
Starting and Stopping Pipeline Manager Manually	1-8
Starting Pipeline Manager	1-9
Stopping Pipeline Manager	1-9
Starting and Stopping Individual Pipelines	1-9
Restarting Pipeline Manager after an Abnormal Shutdown	1-10
Using Perl Scripts to Start and Stop Pipeline Manager	1-10
Starting and Stopping Oracle IMDB Cache DM	1-12

Starting IMDB Cache DM	1-12
Stopping IMDB Cache DM	1-13

2 Monitoring and Maintaining Your BRM System

About Monitoring BRM	2-1
Components Monitored and Controlled by the pin_ctl Utility.....	2-2
About Data Collected by OMF.....	2-2
Using the pin_ctl Utility to Monitor BRM	2-3
Setting Up the pin_ctl Utility.....	2-3
Getting the Status of a Component by Using the pin_ctl Utility	2-4
Clearing Log Files for a Component by Using the pin_ctl Utility	2-4
Getting Diagnostic Data for a Component by Using the pin_ctl Utility	2-4
Diagnostic Data Collected by the pin_ctl Utility	2-5
Configuring the pin_ctl Utility.....	2-6
Customizing the Components Included in “all”	2-6
Customizing pin_ctl.conf for Starting and Stopping Optional Components	2-6
Creating a Custom “all” Parameter List.....	2-8
Customizing the Components List.....	2-8
Customizing the pin_ctl Startup Configuration.....	2-10
Customizing the pin_ctl Utility Environment Variables	2-10
Setting the CTL_SNMP_PATH variable.....	2-12
Setting the pin_ctl Utility Log Level	2-12
Configuring the Start and Stop Validation Settings for pin_ctl	2-12
Customizing snmpset and snmpget Actions.....	2-13
Using Custom pin_ctl Configuration Files.....	2-14
Using the pin_db_alert Utility to Monitor Key Performance Indicators	2-14
KPI Default Behavior	2-15
About KPI Status and Alert Notifications	2-16
About Monitoring KPIs.....	2-17
Setting Up KPI Monitoring	2-17
Specifying Which KPI Data Is Extracted	2-18
Setting Up Email Alert Notifications	2-19
Enabling Database Access	2-20
Monitoring the Size of Audit Tables	2-21
Monitoring the Age of Audit Tables.....	2-21
Monitoring the Age of Events.....	2-21
Monitoring Active Triggers.....	2-22
Monitoring Indexes.....	2-22
Monitoring Stored Procedures.....	2-22
Running the pin_db_alert.pl Utility	2-23
Defining Custom KPIs.....	2-23
Collecting Diagnostic Information by Using RDA	2-23
Installing Remote Diagnostic Agent.....	2-25
Running Remote Diagnostic Agent.....	2-26
Viewing RDA Documentation	2-28
Dumping Business Parameters in XML Format	2-28
Using Logs to Monitor Components	2-30

Types of Log Files	2-30
Log Files for System Components	2-30
Log Files for Applications.....	2-30
Log Files for Client Applications	2-30
Location of Log Files.....	2-31
Default Log File Locations.....	2-31
Changing the Name or Location of a Log File.....	2-31
Setting the Reporting Level for Logging Messages	2-32
Getting Debugging Information from Command-Line Utilities.....	2-32
Dynamically Changing the CM and DM Log Levels	2-33
Setting the Log Level for a Specific Opcode.....	2-33
Recording Opcode Calls in the CM Log File.....	2-34
About Formatting Log Files.....	2-34
Masking Sensitive Data in Log Files	2-34
Maintaining Log Files.....	2-34
Checking the Number and ID of a BRM Process.....	2-35
Dealing with Hung and Looping Processes.....	2-35
Checking for Hung Processes.....	2-35
Checking for Looping Processes	2-35
Stopping a Hung or Looping Process	2-35
Monitoring CM Activity	2-36
Manually Checking the Status of the CM	2-36
Resolving Lost TCP Connections.....	2-36
Enabling Java PCM Clients to Use Operating System TCP/IP Keepalive Parameters	2-37
Setting the CM Log Time Resolution	2-37
Setting a Timeout Value for Requests Sent to the CM	2-37
Configuring Multilevel CM Timeout for Client Requests	2-38
A Short (Suspect) Timeout	2-38
A Long (Failover) Timeout	2-39
Getting Quality of Service Statistics from the CM	2-39
Configuring CM QoS Statistics	2-40
Monitoring DM Activity.....	2-41
Manually Checking the Status of the DM	2-41
Checking the DM Status in flist Format	2-41
Checking the DM Status in a Report Format	2-43
Monitoring DM Shared Memory Usage	2-43
Monitoring DM Transaction Queues	2-44
Monitoring DM Back Ends	2-44
Monitoring DM Front Ends	2-45
Increasing the Level of Reporting for a DM.....	2-45
Using Environment Variables to Set Debug Options	2-46
Editing the Configuration File to Set Debug Options	2-46
Logging the DM Process Time Information for Performance Diagnostics	2-47
Replacing Failed DM Child Processes	2-47
Monitoring Pipeline Manager	2-47
Monitoring Pipeline Manager Memory Usage	2-48
Monitoring Pipeline Manager EDR Throughput	2-48

Getting Recent Pipeline Log File Entries	2-48
Monitoring IMDB Cache DM	2-49
Generating the IMDB Cache DM Core Dump	2-49
Troubleshooting IMDB Cache DM errors	2-49
Getting Opcode Statistics from IMDB Cache DM	2-49
About the Global Transaction System Tables and Views	2-50
About the GLOBAL_TRANS_T Table	2-50
About the GLOBAL_PENDING_TRANS_T Table	2-51
About the DBA_2PC_PENDING View	2-51
Maintaining IMDB Cache DM	2-52
Handling Active IMDB Cache DM Failure	2-53
Handling Active Node Failure.....	2-53
Managing Cache Group Data.....	2-53
Finding and Fixing Global Transaction Errors	2-53
Monitoring Customer Center Activity	2-55
Monitoring Hardware and Operating Systems	2-55
Checking the Version Numbers of Components	2-55
Checking BRM Component Version Numbers.....	2-55
Checking Pipeline Component Version Numbers	2-56
Using the Diagnostic Data Handler to Get OMF Diagnostic Data	2-56
Getting a Snapshot from the Diagnostic Data Handler.....	2-57
About Operations Management Framework	2-57
About Probe Data	2-58
Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components	2-59
About the SNMP Components.....	2-59
Installing SNMP	2-60
Configuring SNMP Components.....	2-60
Problems Using SNMP on Oracle Solaris 10.....	2-60
Enabling SNMP Instrumentation Data Collection	2-61
About the BRM MIB	2-62
About Dynamic Object IDs.....	2-62
About Instance IDs for Table Probes.....	2-63
About the Process Table.....	2-64
About the Registry Table	2-64
Getting and Setting Instrumentation by Using SNMP	2-65
Sample SNMP Input and Output	2-65
snmpGet	2-65
snmpSet	2-65
snmpWalk	2-66
Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data	2-66
Enabling HTTP Display of Instrumentation Data	2-67
Displaying Instrumentation Data in a Web Browser.....	2-68
Customizing the Data Displayed by the HTTP Instrumentation Protocol	2-68

3 Backing Up and Restoring Your BRM System

About Backing Up and Restoring BRM.....	3-1
Backing Up BRM Configuration.....	3-2

Backing Up BRM Files.....	3-2
Backing Up Pipeline Manager Files.....	3-2
Backing Up Rated Event Loader Files.....	3-3
Restoring BRM Configuration.....	3-4

4 Using Configuration Files to Connect and Configure Components

About Configuration and Properties Files	4-1
Configuration Entry Syntax	4-2
Syntax for Configuration Entries	4-2
Syntax for Facilities Module Entries.....	4-2
Preparing for Platform Migration by Using Variables in pin.conf Files	4-3
Locations of Configuration and Properties Files	4-4
File Locations	4-4
Configuring a Shared pin.conf File.....	4-4
Guidelines for Editing Java Properties Files	4-4
Common Properties File Entry Syntax.....	4-5
Connection Entry	4-5
Failover Entry	4-6
Other Properties Entries.....	4-6
About Validating XML Configuration Files	4-6
About Connecting BRM Components	4-7
Guidelines for Database and Port-Number Entries	4-9
Setting Data Manager Attributes	4-10
Connecting a Data Manager to the BRM Database	4-11
Creating Configuration Files for BRM Utilities.....	4-11
Reconfiguring a BRM Component	4-14
Running Non-MTA Utilities in Multischema Systems.....	4-14
Configuring BRM by Using the pin_bus_params Utility	4-15
Retrieving /config/business_params Objects	4-16
Loading /config/business_params Objects.....	4-16

5 Implementing System Security

Using General System-Security Measures	5-1
Basic Security Considerations	5-1
Understanding the BRM Environment.....	5-1
Oracle Security Documentation	5-2
Storage of Passwords in Configuration Files	5-2
Configuring Access to the BRM Database	5-3
Configuring Login Names and Passwords for BRM Access.....	5-3
Configuring the CM to Verify Application Logins with the Service Only	5-4
Configuring the Maximum Number of Invalid Login Attempts.....	5-4
Configuring Applications to Provide Login Information	5-5
Login Information for Java-Based Client Applications.....	5-5
Changing Default Passwords for Java-Based Client Applications.....	5-5
Login Information for Payment Tool	5-5
Login Information for Optional Service Integration Components	5-5

Creating a Customer Service Representative Account	5-6
Changing the Login Name and Password for an Account	5-6
Logging Customer Service Representative Activities	5-6
Setting Up Permissions in BRM Applications	5-7
About Permissioning Center	5-8
About Access to Permissioning Center.....	5-9
About Managing Roles.....	5-9
About Multiple Role Support.....	5-10
About Hierarchical Roles.....	5-10
About Managing Permissions in Permissioning Center	5-10
Managing CSR Passwords.....	5-10
Setting CSR Account Password Status	5-11
Automatic Logout.....	5-11
Changing the Password	5-12
Unlocking a Locked CSR Account	5-12
Setting the Default Password Expiry Duration	5-13
Unlocking the Locked root Account	5-13
How Permission Settings Are Stored	5-14
Permission Types	5-14
Customer Center Permission Type	5-14
Collections Center Permission Types	5-15
Pricing Center Permission Types.....	5-15
Suspense Management Center Permission Types	5-16
Other Client Application Permission Types	5-17
Access Levels	5-17
Setting Up Access to Brands.....	5-17
Protecting BRM Passwords	5-18
Enabling Secure Communication between BRM Components	5-18
Working with SSL/TLS Certificates and Oracle Wallets	5-19
Creating an Oracle Wallet and a Server Certificate	5-20
Using the orapki Utility to Create Oracle Wallets and Certificates.....	5-21
BRM-Supported Cipher Suites.....	5-22
Enabling SSL/TLS in Connection Managers	5-22
Enabling SSL/TLS in Data Managers	5-23
Enabling SSL/TLS for C and C++ PCM Clients	5-24
Enabling SSL/TLS for Java PCM Clients.....	5-25
Enabling SSL for Customer Center Web Start Deployment	5-27
Enabling SSL/TLS for Java Server Processes.....	5-28
Enabling SSL/TLS in Connection Manager Master Processes.....	5-29
Enabling SSL/TLS for Payment Tool	5-30
Enabling SSL/TLS with Custom Applications	5-31
Verifying Server Host Name	5-32
SSL/TLS Client Certificate Authentication	5-32
Creating Debugging Logs for SSL/TLS.....	5-32

6 BRM OMF Instrumented Objects

About the BRM MIB.....	6-1
------------------------	-----

About the Top-Level Components in the MIB	6-1
Transaction Manager Probes	6-1
Log File Probes	6-2
Operations Management Framework Probes	6-2
HTTP Probes	6-2
SNMP Probes	6-3
Diagnostic Data Handler Probe	6-3
Multi-Threaded Framework	6-4
Connection Configurations Probes.....	6-4
Connection Configurations Probes	6-4
Oracle DM Server Probes.....	6-5
Thread Check Manager Probes	6-5
Thread Check Manager Probes.....	6-5
Thread Chart Probes.....	6-5
MTF State Manager Probes.....	6-6
AAA Manager Connection and Number Portability Probes	6-7
Connection Monitor Probes.....	6-7
Number Portability Probes	6-7
Memory Monitor Probes	6-8
Pipeline Statistics Probes	6-8
Throughput Statistics Probes.....	6-8
Last Throughputs Probes.....	6-9
Opcode Latency Probes	6-9
Opcode Plug-In Probes.....	6-9
Last Latency Probes	6-10
Input Probes	6-10
Input Controller Probes.....	6-10
Last Processing Times Probes	6-10
SNMP Object IDs	6-11
MIB Prefix.....	6-11
Transaction Manager OIDs.....	6-12
Log File OIDs	6-12
OMF OIDs	6-12
MTF OIDs.....	6-14
Data Object OIDs.....	6-20
AAA Manager Connection and Number Portability OIDs	6-23
Memory Monitor OIDs.....	6-24
Statistics OIDs.....	6-24
fct Probe OIDs.....	6-25
Input Controller probes.....	6-26
Process Table.....	6-26
Registry Table	6-27

7 Configuring Pipeline Manager

About Configuring Pipeline Manager	7-1
Using Registry Files to Configure Pipeline Manager	7-1
About the Registry File Structure	7-1

About the Registry File Syntax.....	7-3
About the Sample Registry Files.....	7-4
About Configuring Pipelines.....	7-4
About Configuring Function Modules	7-5
About iScripts and iRules	7-7
About Configuring iScripts	7-7
About Configuring iRules	7-8
Configuring Multiple Instances of a Pipeline	7-8
About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands....	7-9
Configuring Multiple Instances of Sequencers, Output Streams, or System Brands.....	7-11
Configuring the Data Pool.....	7-13
Connecting a Module to a Database	7-14
Forcing a Database Reconnection	7-15
Reloading Data into a Pipeline Manager Module	7-15
Using Business Parameter Settings from the BRM Database	7-16
Connecting Pipeline Manager Modules to DAT_PortalConfig.....	7-17
Printing Business Parameter Settings Stored in DAT_PortalConfig Memory	7-17
Refreshing Business Parameter Settings Stored in DAT_PortalConfig Memory	7-18
Connecting a Pipeline Manager Module to Another Module.....	7-19
Configuring Pipeline Buffers	7-19
Using Rogue Wave Buffers.....	7-20
Using Block Transfer Buffers on Solaris Systems.....	7-20
Using Array Buffers on HP-UX Itanium and Solaris Systems	7-21
Using Semaphore Files to Control Pipeline Manager.....	7-23
Updating Configuration Settings during Runtime by Using Semaphore Files	7-23
Configuring Where and How Often the Controller Checks for Semaphore Files.....	7-24
Procedure for Updating Configuration Settings	7-24
Semaphore File Syntax	7-25
Semaphore Error Messages	7-26
Using Events to Start External Programs	7-26
About Mapping Events to Programs	7-27
Controlling External Programs	7-28
About Running External Programs	7-28
Troubleshooting Event Handling	7-28
About Pipeline Manager Transactions	7-28
About the Transaction Manager	7-29
About Cancelling Transactions When a Rollback Occurs	7-30
About Transaction Log Files	7-30
Configuring the Transaction ID Controller.....	7-30
About Storing IDs in Cache.....	7-31
About the Transaction ID State File and Table.....	7-31
Configuring Sequence Checking	7-31
Sequence Numbers in the Header Record.....	7-31
Deciding Whether to Use Sequencers	7-32
About Sequence Checking	7-32
About Sequence Generation	7-32
About Maximum and Minimum Sequence Numbers.....	7-33

About Recycled EDRs.....	7-33
About Sequencer Files and Tables.....	7-34
Sequencer State Files and State Tables.....	7-34
Sequencer Log Files and Log Tables	7-34
Checking and Generating Sequence Numbers.....	7-35
Configuring the NET_EM Module for Real-Time Processing	7-36
Configuring the NET_EM Module.....	7-37
Specifying the Type of NET_EM Opcode Processing	7-37
Configuring the CM to Send Real-Time Requests to the NET_EM Module	7-37
About Pipeline Manager Log Files	7-38
Pipeline Manager Log File Registry Entries.....	7-39
About Error Message Files.....	7-40
About Log File Contents	7-40
Troubleshooting Pipeline Modules.....	7-42
Writing EDR Contents to a Log File	7-43
Using a Semaphore to Write EDR Contents to a File for Debugging.....	7-43
Sample EDR Content Log File	7-43
Using Perl Scripts to Administer Pipeline Manager	7-46

8 Controlling Batch Operations

About the Batch Controller	8-1
Setting Activity Times and Triggers.....	8-1
General Batch Controller Parameters.....	8-1
Connection Parameters	8-1
Time-to-Run Parameters	8-2
Log-File Parameter.....	8-2
Timeout-Limit Parameters.....	8-2
Example of Parameters	8-3
Handler Identification	8-3
Occurrence-Driven Execution	8-4
Timed Execution.....	8-5
Metronomic Execution	8-5
Scheduled Execution	8-7
Starting the Batch Controller	8-8
About SampleHandler	8-8
Copying SampleHandler	8-8
Customizing SampleHandler.....	8-9
Configuring the Batch Controller	8-10
Starting the New Batch Handler	8-10

9 About Connection Pooling

Overview	9-1
Configuring the Connection Pool.....	9-1
Infranet.properties File Connection Pool Parameters.....	9-2
Connection Pool Error Handling.....	9-2
Monitoring Connection Pooling Events	9-3

10 System Administration Utilities and Scripts

load_pin_event_record_map	10-2
partition_utils	10-4
partitioning.pl	10-11
pin_clean_asos	10-13
pin_clean_rsvns	10-15
pin_close_items	10-18
pin_ctl	10-19
pin_db_alert.pl	10-23
pin_purge	10-24
pin_sub_balance_cleanup	10-26
pin_tt_schema_gen	10-27
pin_virtual_gen	10-29
purge_audit_tables.pl	10-31

11 SNMP Utilities

snmpBulk	11-2
snmpDiscover	11-3
snmpGet	11-4
snmpNext	11-5
snmpNextAsync	11-6
snmpPasswd	11-7
snmpSet	11-8
snmpWalk	11-9
snmpWalkThreads	11-10

Part II Administering a High-Availability System

12 Understanding a High-Availability System

About High-Availability BRM Systems	12-1
About the Architecture of a High-Availability BRM System	12-1
About Connection Managers in a High-Availability System	12-3
How CMs Handle Real-Time Pipeline Failure	12-3
How CMs Handle IMDB Cache DM Failure	12-3
Initial Connection Failure	12-3
Session Failure	12-4
About IMDB Cache DMs and Data Stores in a High-Availability System	12-4
How IMDB Cache DMs Fail Over	12-6
About Active Node Failure	12-6
About Active IMDB Cache DM Failure	12-6
Manually Activating a Standby IMDB Cache DM	12-7
How IMDB Cache DMs Handle Data Store Failure	12-7
Initial Connection Failure	12-7
Transaction Failure	12-8
How IMDB Cache DMs Handle Oracle RAC Failure	12-8
Initial Connection Failure	12-9

Session Failure	12-9
About the BRM Database in a High-Availability System	12-10
About Oracle Real Application Clusters and Oracle Clusterware	12-10
About Multischema High-Availability Systems.....	12-11

13 Configuring a High-Availability System

About Setting Up a High-Availability System.....	13-1
Configuring the BRM Database for High Availability	13-3
Setting Up Oracle RAC for Failover in a High-Availability System	13-3
Configuring Oracle Database Services	13-5
Defining Connections to the Oracle Database Services.....	13-5
Connecting IMDB Cache DMs to Oracle RAC Instances for High Availability	13-7
Minimizing Recovery Time of Oracle RAC Instances	13-7
Configuring IMDB Cache Manager for High Availability.....	13-8
Creating and Configuring Active Data Stores	13-10
Initializing an Active Data Store	13-13
Generating the Schema and Load SQL Files for the Active Data Store	13-14
Initializing the Data Store	13-14
Configuring the Cluster for an Active Data Store	13-15
Configuring Standby Data Stores	13-16
Creating Active and Standby Data Store Pairs	13-17
Associating IMDB Cache DM Instances with Data Stores.....	13-18
Making a Data Store Accessible to IMDB Cache DM	13-18
Configuring an IMDB Cache DM Instance for a Data Store.....	13-19
Configuring pin_ctl to Start and Stop IMDB Cache DM	13-20
Configuring Connection Managers for High Availability	13-21
Connecting CMs to IMDB Cache DMs	13-21
Restoring a High-availability System after Failover.....	13-23
Switching Back to the Primary Oracle RAC Instance	13-23
Ensuring All Accounts Are Billed.....	13-23

Part III Improving Performance

14 Improving BRM Performance

Monitoring Performance.....	14-1
Improving Connection Manager Performance.....	14-1
Increasing CM Login Performance.....	14-1
Using CM Proxy to Allow Unauthenticated Log On	14-2
Turning Off Session-Event Logging.....	14-2
Turning Off the Checking of Logons and Passwords	14-3
Load Balancing CMs.....	14-3
Improving Performance for Loading Large Price Lists.....	14-3
Improving Real-Time Rating Performance	14-4
Changing the Precision of Rounded and Calculated Values	14-5
Setting the Interval for Checking for Price List Changes.....	14-5
Setting the Interval for Updating Zone Maps.....	14-5

Filtering the ERAs Considered during Rating and Discounting	14-6
Enabling and Disabling the Caching of Customized Products	14-7
Configuring the Maximum Number of Products and Discounts Cached	14-8
Improving the Performance of your Multithreaded Applications	14-9
Controlling Thread Load on Your Multithreaded Application	14-9
Monitoring the Thread Activity of Your Multithreaded Application	14-10
Logging Noncurrency Events	14-11
Specifying the Number of Connections to CMs	14-11
Setting the CM Time Interval between Opcode Requests	14-12
Using Connection Manager Master Processes	14-12
Sample CMMP Configuration	14-12
Setting Up a CMMP	14-14
Improving Data Manager and Queue Manager Performance	14-14
About Queuing-Based Processes	14-15
Example of Queuing in a Client-to-CM Connection	14-15
Configuring DM Front Ends and Back Ends	14-16
Ratio of Front Ends to Back Ends	14-17
Providing Enough Front-End Connections	14-17
Determining the Required Number of Back Ends	14-18
Determining the Maximum Number of Back Ends Dedicated to Transactions	14-19
Setting the DM Time Interval between Opcode Requests	14-19
Setting How Long the DM Waits for the Background Startup Process to Finish	14-19
Configuring Multiple Pipelines in the DM to Improve Performance	14-20
Setting DM Shared Memory Size	14-20
Determining DM Shared Memory Requirements	14-21
How BRM Allocates Shared Memory for Searches	14-21
Shared Memory Guidelines	14-22
Reducing Resources Used for Search Queries	14-23
Load Balancing DMs	14-23
Optimizing Memory Allocation during Database Searches	14-23
Improving BRM Performance during Database Searches	14-24
Increasing DM CPU Usage	14-24
Examples of DM Configurations	14-24
Improving Interprocess Communication (IPC) Performance	14-25
Sample Configuration Settings for a Multischema System	14-26
Improving Performance by Disabling Unused Features	14-27
Tuning Billing Performance by Using Configuration Files	14-29
Filtering Search Results	14-30
Tuning the Number of Children for Billing Utilities	14-30
Tuning the Account Cache Size for Billing Utilities (fetch_size)	14-31
Tuning the Batch Size for Billing Utilities (per_batch)	14-31
Tuning the Batch Size for the pin_collect Utility	14-31
Specifying the Number of Retries in a Deadlock	14-32
Rearranging Accounts to Improve Billing Performance	14-32
Improving Performance in Retrieving Purchased Offerings for a Bill Unit	14-34
Tuning Memory for Billing	14-35
Additional Issues Related to Billing Performance	14-35

How the Number of Events Affects Billing.....	14-35
Tuning Billing Performance by Using Business Parameters.....	14-36
Improving Performance by Skipping Previous Total Unpaid Bill for Open Item Accounting Type 14-36	
Improving Trial Billing Performance by Enabling General Ledger Collection	14-37
Excluding Searches on Closed Offerings.....	14-38
Excluding Searches on Overridden Products	14-39
Tuning Billing Performance by Using Business Profiles.....	14-40
Improving Performance by Skipping Previous Total for Open Item Accounting Type When Calculating the Current Bill 14-40	
Improving Performance by Using Multiple Item Configurations	14-41
Improving Performance by Skipping Billing-Time Tax Calculation.....	14-42
Tuning Invoicing Performance.....	14-43
Setting the Number of Children for Invoice Utilities	14-43
Tuning the Account Cache Size for Invoice Utilities (fetch_size)	14-44
Setting the Batch Size for Invoice Utilities (per_step).....	14-44
Optimizing Invoicing Performance	14-44
Improving Performance in Retrieving Product Details During Product Purchase.....	14-46
Improving Data Processing Performance	14-47
How Statement-Handle Caching Works	14-47
How to Use the Statement-Handle Cache	14-48
Managing Database Usage	14-48
Rebuilding Indexes.....	14-49
Removing Unused Indexes.....	14-49
BRM Account and Rating Performance Considerations	14-49
Tuning Multithreaded Workloads.....	14-50
CM and DM RAM and Swap Guidelines	14-50
Hardware Guidelines	14-50
Improving Network Performance.....	14-51
Troubleshooting Poor Performance.....	14-51
Low Performance with High CPU Utilization.....	14-51
Low Performance with Low CPU Utilization.....	14-52
Quick Troubleshooting Steps	14-52
About Benchmarking	14-53
BRM Performance Diagnosis Checklist	14-54
Describe the Problem.....	14-55
Describe the Configuration.....	14-55
Hardware Configuration	14-55
Operating System Configuration.....	14-55
BRM Configuration.....	14-55
Network Configuration.....	14-56
Database Server Configuration.....	14-56
Oracle Database Configuration.....	14-56
Describe the Activity	14-57

15 Optimizing Pipeline Manager Performance

Pipeline Manager Optimization Overview.....	15-1
---	------

Key Metrics for Measuring Performance.....	15-1
About Measuring Pipeline Manager Performance	15-2
Information Requirements.....	15-2
Testing Requirements.....	15-2
Optimizing Pipeline Manager.....	15-2
Troubleshooting Pipeline Performance	15-3
Optimizing Function Modules	15-4
Configuring Single-Threaded or Multithreaded Operation	15-4
Reducing Startup Times with Parallel Loading.....	15-5
Assigning Multiple Threads to Process Function Modules.....	15-6
Configuring the DAT_AccountBatch Module Database Connections and Threads.....	15-7
Setting the Hash Map Size for Threads	15-7
Locking Objects during DAT_AccountBatch Processing.....	15-8
Configuring Threads for DAT_BalanceBatch Connections	15-9
Improving Pipeline Manager Startup Performance	15-9
Improving DAT_BalanceBatch Loading Performance	15-9
Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing.....	15-9
Breaking Up Large Nested Subsections in Registry Files.....	15-10
Optimizing a Pipeline by Using Function Pools	15-11
Adding Function Pools.....	15-11
Shifting Modules between Function Pools.....	15-15
Configuring Buffers	15-15
Combining Multiple CDR Files into One Transaction	15-16
Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams	15-17
Configuring Multiple Pipelines.....	15-18
Customizing Flists Sent to a Real-Time Pipeline	15-18
Configuration Object Dot Notation.....	15-19
About the <i>field_list.xml</i> File.....	15-19
Mapping Events to Flists.....	15-21
Usage Example	15-21
Sample Unmodified Flist	15-21
Sample Fields Required by Pipeline Manager.....	15-26
sample.xml File.....	15-27
Measuring System Latencies with Instrumentation	15-32
Using Instrumentation to Collect Module Performance Statistics.....	15-33
Viewing Instrumentation Testing Results.....	15-33
Sample Log File.....	15-34
Optimizing the DAT_USC_Map Module	15-34
About Precompiling USC Mapping Rules	15-35
About Filtering Mapping Rules	15-35
Configuring the DAT_USC_Map Module for Startup Performance.....	15-36
Increasing the Number of Threads Used to Load Mapping Data	15-36
Precompiling Usage Scenario Mapping Data	15-36
Filtering the Mapping Data to Compile and Load.....	15-36
Using Semaphores	15-37
Other Pipeline Manager Monitoring Tools	15-37
Viewing the Pipeline Log.....	15-37

Configuring Buffer Size Polling	15-38
OS-Specific Pipeline Manager Monitoring Tools	15-38
Solaris Monitoring Tools	15-38
Displaying Thread Details	15-38
Dumping the Call Stack of an Active Process	15-39
Identifying Thread Functions	15-39
HP-UX IA64 Monitoring Tools	15-39
Linux Monitoring Tools	15-40
AIX Monitoring Tools	15-40

16 Optimizing BRM for Prepaid and Postpaid Convergence

About Convergent BRM Systems	16-1
About Cache Residency	16-1
How Cache Residency Is Determined	16-2
About Setting Up Business Profiles for Cache Residency Distinction	16-4
How BRM Caches Objects in a Convergent Rating System	16-4
About Changing the Cache Type of an Object	16-5
Configuring BRM for Cache Residency Distinction	16-7
Enabling Cache Residency Distinction	16-7
Assigning Objects a Cache Residency Value	16-8
Setting Up Cache Residency Validation Rules	16-8
Changing the Default Business Profile	16-9
Overriding the Default Business Profile	16-10
About Selective Account Loading	16-10
Configuring Pipeline Manager for Selective Account Loading	16-11
Enabling Selective Account Loading	16-11
Configuring Pipeline Manager to Process Prepaid CDRs	16-12
Customizing Cache Residency Distinction	16-12
Configuring Nonreference Objects for Cache Residency Distinction	16-13

Part IV Troubleshooting BRM

17 Resolving Problems in Your BRM System

General Checklist for Resolving Problems with BRM	17-1
Using Error Logs to Troubleshoot BRM	17-2
Finding Error Log Files	17-2
Understanding Error-Message Syntax	17-2
Resolving Clusters of Error Messages	17-3
Logging External User Information in Error Logs	17-3
Interpreting Error Messages	17-4
Example 1: Failure of a Client Application	17-4
Example 2: Problem with Customer Center	17-5
Example 3: Getting More Information from Error Numbers	17-5
Example 4: Getting More Information about Oracle Errors	17-5
Diagnosing Some Common Problems with BRM	17-6
Problems Starting BRM Components	17-6

Problem: Bad Bind, Error 13.....	17-6
Problem: Bad Bind, Error 125.....	17-6
Problem: Cannot Connect to Oracle Database.....	17-7
Problem: ORA-01502: Index 'PINPAP.I_EVENT_ITEM_OBJ__ID' or Partition of Such Index Is in Unusable State	17-7
Problems Stopping BRM Components	17-8
Problem: No Permission to Stop the Component	17-8
Problem: No pid File	17-8
Problems Connecting to BRM	17-8
Problem: Cannot Connect to the Database.....	17-8
Problem: Cannot Connect to the CM	17-8
Problem: CM Cannot Connect to a DM.....	17-9
Problems with Deadlocking	17-9
Problem: BRM “Hangs” or Oracle Deadlocks	17-10
Problem: dm_oracle Cannot Connect to the Oracle Database	17-10
Problems with Memory Management	17-10
Problem: Out of Memory	17-10
Problem: Java Out of Memory Error.....	17-13
Problem: Memory Problems with the Oracle DM	17-13
Problems Running Billing.....	17-14
Problem: Billing Daemons Are Running, but Nothing Happens	17-14
Problem: High CPU Usage for the Number of Accounts Processed.....	17-14
Problems Creating Accounts	17-14
Problem: fm_delivery_mail_sendmsgs Error Reported in the CM Log File.....	17-14
Problems Loading Configuration Objects	17-15
Problem: Failed to create XML context in isXsltExists, error [266].....	17-15
Getting Help with BRM Problems	17-15
Before You Contact Technical Support	17-15
Reporting Problems	17-15

18 Reference Guide to BRM Error Codes

Interpreting BRM Error Codes	18-1
BRM Error Locations	18-1
BRM Error Classes	18-2
BRM Error Codes.....	18-3

19 Pipeline Manager Error Messages

Pipeline Framework Error Messages.....	19-1
Pipeline Manager Module Error Messages	19-15
DAT_AccountBatch	19-15
DAT_AccountRealtime	19-19
DAT_BalanceBatch.....	19-20
DAT_BalanceRealtime.....	19-21
DAT_ConnectionManager.....	19-21
DAT_ConnectionPool.....	19-22
DAT_Currency	19-22
DAT_Discount	19-22

DAT_ExchangeRate	19-23
DAT_InterConnect.....	19-23
DAT_ItemAssign.....	19-24
DAT_Listener.....	19-24
DAT_ModelSelector.....	19-25
DAT_NumberPortability	19-26
DAT_PortalConfig	19-26
DAT_Price	19-26
DAT_Rateplan	19-27
DAT_Recycle.....	19-27
DAT_ResubmitBatch	19-27
DAT_ScenarioReader	19-28
DAT_USC_Map.....	19-28
DAT_Zone.....	19-29
EXT_InEasyDB.....	19-29
EXT_PipelineDispatcher	19-29
FCT_Account	19-29
FCT_AccountRouter	19-30
FCT_AggreGate.....	19-30
FCT_APN_Map	19-30
FCT_ApplyBalance	19-31
FCT_BatchSuspense.....	19-31
FCT_BillingRecord	19-32
FCT_CallAssembling.....	19-32
FCT_CarrierIcRating.....	19-34
FCT_CiberOcc.....	19-34
FCT_CliMapping.....	19-35
FCT_CreditLimitCheck	19-35
FCT_CustomerRating.....	19-35
FCT_DataDump	19-35
FCT_Discard	19-35
FCT_Discount	19-35
FCT_DroppedCall.....	19-37
FCT_DuplicateCheck.....	19-37
FCT_EnhancedSplitting	19-38
FCT_ExchangeRate	19-38
FCT_Filter_Set	19-38
FCT_IRules.....	19-39
FCT_IScriptPlugIn.....	19-39
FCT_ItemAssign.....	19-39
FCT_MainRating	19-39
FCT_Opcode	19-40
FCT_PreRating.....	19-40
FCT_PreSuspense.....	19-41
FCT_RateAdjust	19-41
FCT_Reject.....	19-41
FCT_Rounding	19-41

FCT_SegZoneNoCust	19-42
FCT_Suspense	19-42
FCT_Timer	19-42
FCT_TriggerBill	19-42
FCT_UoM_Map	19-42
FCT_UsageClassMap	19-43
FCT_USC_Map	19-43
INP_GenericStream	19-43
INP_Realtime	19-43
INP_Recycle	19-45
NET_EM	19-46
OUT_DB	19-46
OUT_GenericStream	19-47
OUT_Realtime	19-47
Pipeline Utility Error Messages	19-47
LoadIFWConfig	19-47
OMF Error Messages	19-47

Part V Managing IMDB Cache-Enabled Systems

20 Using Oracle IMDB Cache Manager

About Oracle IMDB Cache Manager	20-1
About Caching BRM Objects in Memory	20-1
About BRM Cache Groups	20-1
Guidelines for Setting Cache Group Size	20-2
About Loading BRM Objects into Oracle IMDB Cache	20-2
About Loading Migrated Accounts into the Cache Groups	20-3
About Managing Data in Oracle IMDB Cache	20-3
Managing Fast-Growing Tables	20-3
About Purging Expired Reservation and Active-Session Objects	20-3
About Purging Closed Bills, Items, Journals, and Expired Subbalances	20-3
How Objects Are Stored in Oracle IMDB Cache-Enabled Systems	20-4
About the Residency Type and Oracle IMDB Data Manager	20-4
About Storing Usage Events	20-4
Using Oracle Functions or Procedures in an SQL Statement	20-4
About Searching for Usage Events in Oracle IMDB Cache-Enabled Systems	20-5
About Committing Transactions in Both Oracle IMDB Cache and the BRM Database	20-5
About Recovering from Oracle Global Transaction Errors	20-6
About Manually Fixing Oracle Global Transaction Failures	20-6
Using the Oracle IMDB Cache Grid to Partition Data	20-7
About Logical Partitioning in High Availability Systems	20-7
How Accounts Are Assigned to the Logical Partition	20-9
About Finding Data in Logical Partitions	20-10
About Performing Searches in Logical Partitions	20-10
About Retrieving Data from Logical Partitions	20-11
How IMDB Cache DM Connects to Oracle IMDB Cache	20-12
About the AAA Flow in a BRM System with IMDB Cache Manager	20-12

21 Installing IMDB Cache Manager

About Setting Up a BRM System with IMDB Cache Manager	21-1
About Setting Up IMDB Cache Manager for a Basic BRM System	21-1
About Setting Up IMDB Cache Manager in a Multischema System	21-2
Hardware and Software Requirements.....	21-3
Installing and Configuring a BRM System with IMDB Cache Manager	21-4
Installing IMDB Cache Manager.....	21-4
Obtaining Information Needed for Installing IMDB Cache Manager	21-5
Installing IMDB Cache Manager	21-5
Running the pin_setup Script.....	21-6
Granting Privileges to the Global Transaction Tables.....	21-7
Creating the Data Store in Oracle IMDB Cache.....	21-7
Creating Data Stores for a Basic BRM System	21-7
Creating Data Stores for Logical Partitioning.....	21-10
Initializing Your Data Stores in Oracle IMDB Cache.....	21-13
Connecting Your Data Stores to the BRM Database	21-14
Connecting IMDB Cache DM to Your Data Stores	21-14
Connecting the IMDB Cache DM to Your Data Store	21-14
Connecting Each IMDB Cache DM Instance to a Data Store	21-15
Connecting the CM to IMDB Cache DM	21-17
About Installing Optional Components.....	21-18
Uninstalling IMDB Cache Manager.....	21-18

22 Configuring IMDB Cache Manager

IMDB Cache Manager Configuration Tasks	22-1
Configuring IMDB Cache Manager for Transaction Consistency	22-1
Enabling the Transaction Consistency Feature.....	22-1
Specifying How Long to Keep Transactions Active after an Error.....	22-2
Searching for Events in Both Oracle IMDB Cache and the BRM Database.....	22-3
Enabling Union Searches for Events	22-3
Specifying the Timeout Value for Batch Polling Operations.....	22-4
Customizing External Applications for Real-Time Union Searches.....	22-4
Customizing External Applications for Batch Polling Operations	22-4
FM_UTILS_POLL_TT.....	22-4
Customizing External Applications for Logical Partitions	22-5
Retrieving Data Across Logical Partitions.....	22-5
Searching for Data Across Logical Partitions.....	22-6
Configuring Your Event Tables for BRM Reports.....	22-7
Configuring How to Store Reservations.....	22-8
Setting the Stacksize Limit.....	22-8
Setting Database Schema Status for Oracle IMDB Cache Systems	22-8
Setting the Database Schema Priority for Oracle IMDB Cache Systems.....	22-9

23 Generating the BRM Cache Group Schema

About Generating the BRM Cache Group Schema	23-1
Creating and Initializing Your Cache Group Schema	23-1

Configuring the pin_tt_schema_gen.values File	23-2
Defining Cache Group Information	23-3
Defining Cache Group Information for Special Cases	23-4
Setting On Delete Cascade.....	23-5
Defining the Cache Group Aging Policy	23-5
Defining Transient Local Tables	23-6
Supporting Oracle IMDB Cache Data Types	23-6
Generating the Load SQL Script	23-6
Defining the Logical Partition Database Number.....	23-7
Generating Your Schema and Load SQL Scripts.....	23-7
Generating Scripts for a Basic BRM System.....	23-8
Generating Scripts for a BRM System with Logical Partitioning.....	23-8

24 Customizing IMDB Cache Manager

About Customizing the Cache Groups	24-1
Customizing the Default BRM Cache Groups	24-1
About Extending Tables in the Default BRM Cache Groups	24-2
Creating a Custom Cache Group	24-2
Creating Custom Fields and Storable Classes	24-3
Assigning Custom Objects a Residency Value	24-3
About Extending Event Type Storable Classes.....	24-3
Logical Partitioning and POID DB Translation	24-3
PIN_GET_SCHEMA_NO.....	24-3
Syntax	24-4
Parameters.....	24-4
Return Values	24-4
Error Handling	24-4

25 Migrating Data to an Oracle IMDB Cache-Enabled BRM System

About Migrating Data to an Oracle IMDB Cache-Enabled BRM System	25-1
BRM Processes That Cannot Coexist with IMDB Cache Manager	25-2
How IMDB Cache Manager Migrates Your Existing BRM Accounts	25-2
How IMDB Cache Manager Handles Subscriber Distribution for Account Migration.....	25-3
How IMDB Cache Manager Handles POID Fields.....	25-3
Using the Same Logical Partition for Hierarchical Accounts	25-3
Overview of the Migration Process	25-4
Configuration Entries Required for Migration.....	25-4
Preparing Accounts for Distribution into IMDB Cache Logical Partitions	25-5
Migrating a BRM TIMOS Environment to an IMDB Cache Environment	25-6
Loading Subscriber Data into the Oracle IMDB Cache Data Stores	25-6
Changes to BRM after Installing IMDB Cache Manager.....	25-6
Opcode Changes	25-7
Utility Changes.....	25-7

Part VI Partitioning and Managing BRM Tables

26 Partitioning Tables

About Partitioning	26-1
Overview of Enabling Partitioning during Installation	26-2
About Partitioning Schemes	26-3
About Nonpurgeable Events and Items	26-4
Associating Items with Nonpurgeable Events	26-4
About Objects Stored in partition_last and partition_last_pin.....	26-6
About the Default Partitioning Scheme	26-6
If You Do Not Create Default Partitions.....	26-7
Conversion Partitioning Scheme	26-7
About Partitions for Delayed Events.....	26-7
Overview of Partitioning Schemes	26-8
About Managing Partitions	26-10
About Purging Objects	26-11
About Purging Objects by Removing Partitions	26-11
About Purging Objects without Removing Partitions	26-12
About Running the partition_utils Utility	26-12
Partition Naming Convention.....	26-13
Running the partition_utils Utility in Test Mode.....	26-13
Configuring a Database Connection	26-13
Improving Performance When Using partition_utils.....	26-13
Restarting partition_utils	26-14
Adding Partitions	26-14
Enabling Delayed-Event Partitioning	26-17
Disabling Delayed-Event Partitioning	26-18
Updating Partitions	26-18
Purging Objects by Removing Partitions	26-20
Purging Objects without Removing Partitions	26-22
Finding the Maximum POID for a Date	26-23
Customizing Partition Limitations	26-23
Customizing the List of Events and Items Stored in partition_historic	26-24

27 Converting Nonpartitioned Classes to Partitioned Classes

About Converting Nonpartitioned Classes to Partitioned Classes	27-1
Converting Nonpartitioned Classes to Partitioned Classes	27-1
Increasing Disk Space for Tables and Indexes	27-2
Installing the Partitioning Package.....	27-2
(Optional) Reconfiguring the Parameters.....	27-3
Merging the pin_setup.values File	27-4
Backing Up Your BRM Database	27-4
Running the Partitioning Conversion Scripts	27-4
Adding Purgeable Partitions to Tables	27-4
Restarting BRM.....	27-4
About the Conversion Scripts and Files	27-4
Converting Additional Nonpartitioned Classes to Partitioned Classes	27-5

28 About Purging Data

About Purging Database Objects	28-1
Objects Purged by Default	28-1
About Purging BRM Event Objects	28-2
Event Objects That Have a Balance Impact	28-2
Event Objects That Do Not Have a Balance Impact	28-2
Event and Item Objects Stored in partition_historic	28-3
Impact of Purging Event Objects	28-4
Billing Event Objects	28-4
Accounts Receivable Event Objects	28-8
Delayed Event Objects	28-10
Group Event Objects	28-10
Sharing Group Event Objects	28-11
Session Event Objects	28-11
Auditing Event Objects	28-11
Enabling Open Items to Be Purged	28-12
Closing Open Item Objects Processed in Past Billing Cycles	28-13
About Purging Account Subbalances	28-13

29 Generating Virtual Columns on Event Tables

About Generating Virtual Columns on Event Tables	29-1
Generating Virtual Columns on Event Tables	29-2
Viewing Tasks for Generating Virtual Columns	29-3
Viewing and then Running Virtual-Column Tasks	29-4
Viewing Virtual-Column Task Details	29-5
Exporting a BRM Schema with Virtual Columns	29-5

Part VII Managing a Multischema System

30 Managing a Multischema System

About Multischema Systems	30-1
Using Pipeline Manager with Multiple Database Schemas	30-2
Converting a Single-Schema System to a Multischema System	30-2
Preparing to Manage a Multischema System	30-2
Adding a BRM Installation Machine to a Multischema Environment	30-2
Adding Database Schemas to a Multischema System	30-3
Configuring the pin_multidb.conf File on the Primary Installation Machine	30-4
Setting Database Schema Status	30-5
Setting Database Schema Priorities	30-6
Creating Custom Tables That Are Available to All Database Schemas	30-7
Synchronizing Database Schema Data Dictionaries	30-8
Synchronizing the Database Schema /uniqueness Objects	30-8
Changing the Interval for Updating the Distribution Table	30-9

31 Multischema Utilities

load_config_dist	31-2
------------------------	------

load_pin_uniqueness.....	31-3
pin_config_distribution	31-4
pin_mta_monitor	31-5
pin_multidb.....	31-7

Part VIII Migrating Accounts

32 Understanding Account Migration

About Account Migration.....	32-1
When to Migrate Accounts	32-2
About Account Migration Processes.....	32-2
Account Migration Involving Only the BRM Database	32-2
Account Migration Involving a BRM Database and an Oracle IMDB Cache	32-2
Account Migration Between the Logical Partitions in One IMDB Cache Grid	32-2
Scheduling Account Migration	32-3
About the AMM Application.....	32-3
How AMM Works	32-3
How AMM Works with Oracle IMDB Cache Manager in the BRM Environment.....	32-3
About Migrating Accounts When the Pipeline Manager Is Online.....	32-3
Deleting Migrated Objects	32-4
About Migrating Hierarchical, Sponsorship, and Resource Sharing Accounts	32-4
About Searching for Member and Nongroup Member Accounts	32-4
About Account Groups	32-5
About Migrating Account Groups	32-5
AMM Process Overview	32-6
About the Account Search Configuration File.....	32-7
About the pin_amt Utility.....	32-7
About the AMM Controller.....	32-8
How AMM Controller Processes Batches with Nongroup Members.....	32-8
How AMM Controller Processes Batches with Account Group Members	32-9
About the AMM Mover	32-9
About Distributed Transactions.....	32-10
Account Migration Restrictions.....	32-10
Account Activity Prevented during Account Migration.....	32-11
Do Not Rerate Events during Account Migration	32-11
Do Not Alter Account Group Members	32-11
Migration Prevented during Account Activity.....	32-11
Brand and Remittance Accounts Not Migrated	32-12
Unique POIDs Required across All Database Schemas.....	32-12
Transient Objects Are Not Migrated	32-12
Some Client Applications May Fail during Account Migration	32-12
AMM Does Not Support Some BRM Components.....	32-12
Using BRM Reports after Migration.....	32-12
About Using Multiple AMM Controllers	32-13
Account Migration Performance	32-13
About AMM Job Management Tables.....	32-14

About Job Status Flags	32-14
About Batch Status Flags	32-15
About Group Status Flags	32-15

33 Installing and Configuring BRM for Account Migration

System Requirements	33-1
General Software Requirements	33-1
Pipeline Manager Requirements	33-2
Software Requirements for Oracle IMDB Data Manager	33-2
Software Requirements for Account Migration Manager	33-2
Installing AMM	33-3
Configuring All Primary and Secondary Database Schemas	33-3
Installing the AMM Software on the Primary Installation Machine	33-5
Configuring Your Oracle DM to Check for Invalid Objects	33-7
Connecting AMM to Your Database Schemas	33-7
Configuring the AMM Infranet.properties File	33-7
Configuring Database and AMM Mover Information for Multischema Systems	33-8
Configuring AMM Controller Definitions	33-8
Specifying Schema Alias Information for Multischema Systems	33-9
Enabling the Unloading of Cache Groups from the Source Oracle IMDB Cache	33-9
Specifying Access Information for the Source Oracle IMDB Cache	33-10
AMM Infranet.properties File Parameters	33-10
Sample Infranet.properties File	33-12
Configuring the TimesTen JDBC Driver Jar file for BRM	33-14
Configuring the load_pin_uniqueness Utility for Oracle IMDB Cache	33-14
What's Next?	33-15
Configuring AMM for Additional Database Schemas	33-15
Configuring AMM for New Custom Tables	33-15
Tuning Your Database for Optimal Account Migration Performance	33-16

34 Migrating Accounts with the Pipeline Manager Running

About Migrating Accounts When Pipeline Manager Is Online	34-1
How AMM Interacts with Your Pipelines during Account Migration	34-2
About Waiting before Migrating Accounts	34-3
About Starting Multiple Jobs Concurrently	34-3
About Notifying the Pipelines about Account Migration	34-3
About AMM Business Events	34-4
About Sending AMM Business Events to the Pipelines	34-4
About Notifying AMM about EDR Processing	34-5
About Acknowledgment Events	34-5
About Sending Acknowledgments to AMM	34-6
About the Account Router Instance of the Pipeline Manager	34-6
About Suspending Call Records	34-7
About Reprocessing Suspended Call Records	34-8
Configuring Your System to Migrate Accounts When the Pipeline Manager Is Running ...	34-9
Configuring Your Account-Router Pipeline Manager	34-9
Configuring Your Routing Pipeline	34-9

Configuring Your Pre-recycling Pipeline	34-10
Configuring Your Resuspending Pipeline	34-11
Configuring the Data Pool.....	34-11
Configuring BRM to Handle Suspended EDRs.....	34-12
Configuring AMM to Send Business Events to Your Pipelines	34-13
Connecting AMM to the Primary CM	34-13
Configuring Account Synchronization.....	34-13
Configuring Your Pipelines to Dequeue AMM Business Events	34-15
Configuring Your Pipelines to Send Acknowledgments to AMM.....	34-15
Creating the Acknowledgment Queue	34-15
Connecting AMM Directly to Your Acknowledgment Queue	34-16
Configuring Your Pipelines to Send Acknowledgment Events.....	34-16
 35 Using Account Migration Manager	
Overview of Account Migration Tasks	35-1
Verifying Your System Configuration When Oracle IMDB Cache Is Used.....	35-2
Creating the Account Search Configuration File	35-2
Sample Account Search Configuration File	35-4
Submitting the Account Search File	35-4
Enabling Migration Jobs in the Queue	35-5
Starting the AMM Controller	35-5
Monitoring the AMM Controller	35-5
Checking the AMM Controller Status.....	35-5
Checking the AMM Controller Log File	35-6
Monitoring the AMM Controller in Real Time.....	35-6
Monitoring Account Migration	35-6
Monitoring Job Status	35-6
Checking Job Details.....	35-7
Checking Account Group Details.....	35-7
Handling Account Migration Failures	35-8
Finding Debugging Information.....	35-8
Reprocessing Failed Batches.....	35-9
Purging Migrated Objects from the Source Database Schema	35-9
Deleting Jobs from the Source Database Schema.....	35-9
Stopping the AMM Controller	35-9
Pausing and Resuming Account Migration	35-10
Loading Destination Oracle IMDB Cache with Data from BRM Database	35-10
Automating Account Migration	35-11
 36 Migrating Accounts within an Oracle IMDB Cache Grid	
About Account Migration within an Oracle IMDB Cache Grid.....	36-1
When to Migrate Accounts within an Oracle IMDB Cache Grid.....	36-1
How BRM Migrates Account Data within an Oracle IMDB Cache Grid.....	36-2
About the pin_amt_tt Utility	36-2
About the Transient Objects Migrated by the pin_amt_tt Utility.....	36-2
About Account Synchronization with Pipeline Manager after the Migration	36-3

About the Message Enqueued for the Pipeline	36-3
System and Configuration Requirements	36-3
Guidelines for Migrating Accounts with the pin_amt_tt Utility	36-4
About Account Migration Tasks	36-4
Overview of Migration Procedure	36-4
Configuring the pin_amt_tt Utility	36-5
Providing the Account Selection Criteria	36-5
Configuration File Parameters	36-5
Running the pin_amt_tt Utility	36-8
Monitoring Account Migration	36-8
Handling Account Migration Failures	36-8
Migrating the Transient Objects Manually	36-9
Cleaning Up the Oracle IMDB Cache after Migration	36-9
 37 Modifying Applications to Work with AMM	
Modifying Custom Client Applications for AMM	37-1
Modifying Custom BRM Reports for AMM	37-2
AMM Return Codes and Messages	37-2
 38 Modifying the Account Migration Manager	
Creating Custom Account Search Criteria	38-1
Creating a Search Template	38-1
Sample Search Template	38-2
Adding New Entries to the Account Search Configuration File	38-2
Sample Account Search Configuration File	38-2
Implementing and Compiling the Conversion Interface	38-3
Sample Class Implementing Conversion Interface	38-3
Verifying Your Search Criteria	38-3
Verifying That the Search Criteria Creates Valid SQL Statements	38-4
Verifying That the Search Criteria Finds Correct Accounts	38-4
 39 AMM Entity Relationship Diagram	
 40 Account Migration Utilities	
pin_amt	40-2
pin_amt_test	40-5
pin_amt_tt	40-7

Part IX Running Business Operations

41 Using Business Operations Center

About Using Business Operations Center	41-1
About Performing Operations in Business Operations Center	41-1
Enabling Secure Communication for the pin_job_executor Utility	41-2
Rendering Invoices in a Third-Party Invoice Application	41-3

About Generating Metrics to Display in Business Operations Center	41-3
Generating Business Metrics Data	41-4
Creating Indexes to Improve Performance While Generating Metrics Data	41-5
42 Business Operations Center Utilities	
pin_generate_analytics	42-2
pin_job_executor	42-4
Part X Configuration File Reference	
43 Business Logic pin.conf Reference	
Accounts Receivable pin.conf Entries	43-1
Billing pin.conf Entries	43-2
Collections pin.conf Entries	43-5
Customer Management pin.conf Entries	43-7
Discounting pin.conf Entries	43-7
General Ledger pin.conf Entries	43-9
Invoicing pin.conf Entries	43-9
Payments pin.conf Entries	43-11
Pricing and Rating pin.conf Entries	43-14
Publication pin.conf Entries	43-16
Registration pin.conf Entries	43-16
Revenue Assurance pin.conf Entries	43-17
Service Lifecycle Management pin.conf Entries	43-18
Services Framework pin.conf Entries	43-18
Tax Calculation pin.conf Entries	43-20
44 System Administration pin.conf Reference	
Connection Manager (CM) pin.conf Entries	44-1
Data Manager (DM) pin.conf Entries	44-3
EAI Manager pin.conf Entries	44-6
In-Memory Database (IMDB) Cache DM pin.conf Entries	44-8
Multithreaded Application (MTA) Framework pin.conf Entries	44-8
45 business_params Reference	
Accounts Receivable business_params Entries	45-1
Activity business_params Entries	45-3
Billing business_params Entries	45-3
Customer Management business_params Entries	45-8
General Ledger business_params Entries	45-9
Invoicing business_params Entries	45-9
Pricing and Rating business_params Entries	45-10
Selective Account Loading business_params Entries	45-12
Subscription business_params Entries	45-12
System Administration business_params Entries	45-13

Preface

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

Audience

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

Accessing Oracle Communications Documentation

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

<http://docs.oracle.com>

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

<https://edelivery.oracle.com>

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Document Revision History

The following table lists the revision history for this document:

Version	Date	Description
E16719-01	November 2011	Initial release.

Version	Date	Description
E16719-02	March 2012	Documentation updates. <ul style="list-style-type: none"> Added documentation for the following: Basic Security Considerations Understanding the BRM Environment Oracle Security Documentation
E16719-03	May 2012	Documentation updates for BRM 7.5 Patch Set 1. <ul style="list-style-type: none"> Made minor formatting and text changes. Added the following items for custom service life cycles: Service Lifecycle Management pin.conf Entries section SubscriberLifeCycle business parameter
E16719-04	August 2012	Documentation updates for BRM 7.5 Patch Set 2. <ul style="list-style-type: none"> Added documentation for the following: Account migration when your BRM environment includes the Oracle In-Memory Database (IMDB) Cache Manager in "Migrating Accounts". Added a program column to all tables in "Business Logic pin.conf Reference". Added documentation for "Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams".
E16719-05	December 2012	Documentation updates for BRM 7.5 Patch Set 3. <ul style="list-style-type: none"> Updated the following sections for policy-driven charging: Improving Performance for Loading Large Price Lists Pricing and Rating pin.conf Entries Added documentation for generating virtual columns on event tables: Generating Virtual Columns on Event Tables pin_virtual_gen Added an entry for dd_write_enable_types in "System Administration pin.conf Reference". Added documentation for Pipeline Manager startup performance: Improving DAT_BalanceBatch Loading Performance Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing Updated the "Connection Manager (CM) pin.conf Entries" section. Added an entry for TransferScheduledActions in "Subscription business_params Entries". Added an entry for publish_migrated_objects in "AMM Infranet.properties File Parameters".

Version	Date	Description
E16719-06	March 2013	<p>Documentation updates for BRM 7.5 Patch Set 4.</p> <ul style="list-style-type: none"> ■ Added the "Using Array Buffers on HP-UX Itanium and Solaris Systems" section. ■ Updated the "Manually Checking the Status of the DM" section. ■ Added an entry for SplitSponsorItemsByMember in "Billing business_params Entries". ■ Updated the tables in the following chapters: Business Logic pin.conf Reference System Administration pin.conf Reference ■ Added documentation for configuring multiple instances of sequencers, output streams, or system brands for multiple roaming partners: About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands ■ Replaced multidatabase information with multischema information. See especially "Managing a Multischema System".
E16719-07	July 2013	<p>Documentation updates for BRM 7.5 Patch Set 5.</p> <ul style="list-style-type: none"> ■ Made minor formatting and text changes. ■ Added AMM multischema updates in the following chapters: Using Account Migration Manager Modifying Applications to Work with AMM Modifying the Account Migration Manager Account Migration Utilities
E16719-08	August 2013	<p>On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5.</p> <p>Documentation added for HP-UX IA64.</p>
E16719-09	October 2013	<p>Documentation updates for BRM 7.5 Patch Set 6.</p> <ul style="list-style-type: none"> ■ Made minor formatting and text changes. ■ Added an entry for NonCurrencyResourceJournaling business parameter in "Billing business_params Entries". ■ Added an entry for EnablePasswordRestriction business parameter in "Customer Management business_params Entries". ■ Added an entry for ECERating business parameter in "Pricing and Rating business_params Entries". ■ Added "Enabling Secure Communication between BRM Components".
E16719-10	February 2014	<p>Documentation updates for BRM 7.5 Patch Set 7.</p> <ul style="list-style-type: none"> ■ Made minor formatting and text changes. ■ Added "Enabling SSL/TLS for Payment Tool".

Version	Date	Description
E16719-11	May 2014	<p>Documentation updates for BRM 7.5 Patch Set 8.</p> <ul style="list-style-type: none"> ■ Added information about extended architecture (XA) transactions to "About the DBA_2PC_PENDING View". ■ Added BRM error codes 122–143 to "BRM Error Codes". ■ Added the dm_xa_trans_timeout_in_secs entry to "DM pin.conf Entries".
E16719-12	August 2014	<p>Documentation updates for BRM 7.5 Patch Set 9.</p> <ul style="list-style-type: none"> ■ Added an entry for CreateTwoEventsInFirstCycle business parameter in "Subscription business_params Entries". ■ Documentation added for RADIUS Manager. ■ Updated the "Using Logs to Monitor Components" section. ■ Updated the "BRM-Supported Cipher Suites" section. ■ Updated the "Creating an Oracle Wallet and a Server Certificate" section. ■ Updated the "Enabling SSL/TLS in Connection Managers" section.
E16719-13	October 2014	<p>Documentation updates for BRM 7.5 Patch Set 10.</p> <ul style="list-style-type: none"> ■ Updated the following sections: <ul style="list-style-type: none"> Table 4–5, "MTA Utilities pin.conf Entries" Table 5–1, "Collections Center Permission Types" Enabling SSL/TLS for Java PCM Clients Table 5–6, "BRM Client Infranet.properties File Default Locations" Enabling SSL/TLS for Payment Tool Using Business Parameter Settings from the BRM Database Adding Database Schemas to a Multischema System General Ledger business_params Entries
E16719-14	January 2015	<p>Documentation updates for BRM 7.5 Patch Set 11.</p> <ul style="list-style-type: none"> ■ Made minor formatting and text changes. ■ Added an entry for ApplyDeferredTaxDuringRerating business parameter in "Pricing and Rating business_params Entries". ■ Updated the following tables: <ul style="list-style-type: none"> Table 4–5, "MTA Utilities pin.conf Entries" Table 44–2, "DM pin.conf Entries" Table 44–5, "MTA Framework pin.conf Entries"

Version	Date	Description
E16719-15	June 2015	<p>Documentation updates for BRM 7.5 Patch Set 12.</p> <ul style="list-style-type: none"> ■ Made minor formatting and text changes. ■ Added the following sections: <ul style="list-style-type: none"> Improving Performance by Using Multiple Item Configurations Improving Performance by Skipping Billing-Time Tax Calculation Enabling Open Items to Be Purged pin_close_items ■ Updated the following sections: <ul style="list-style-type: none"> Enabling Secure Communication between BRM Components Logging Customer Service Representative Activities pin_multidb Setting DM Shared Memory Size ■ Removed the UseActualBilledTimeForGLReport business parameter from "General Ledger business_params Entries".
E16720-16	August 2015	<p>Documentation updates for BRM 7.5 Maintenance Patch Set 1.</p> <ul style="list-style-type: none"> ■ Updated the partitioning and purging information in the following chapters: <ul style="list-style-type: none"> Chapter 26, "Partitioning Tables" Chapter 27, "Converting Nonpartitioned Classes to Partitioned Classes" Chapter 28, "About Purging Data" Chapter 10, "System Administration Utilities and Scripts" ■ Added an entry for OfferEligibilitySelectionMode business parameter in "Pricing and Rating business_params Entries". ■ Added an entry for SegregateJournalsByGLPeriod business parameter in "General Ledger business_params Entries". ■ Removed Taxware information in the following chapters: <ul style="list-style-type: none"> Chapter 1, "Starting and Stopping the BRM System" Chapter 2, "Monitoring and Maintaining Your BRM System" Chapter 4, "Using Configuration Files to Connect and Configure Components" Chapter 43, "Business Logic pin.conf Reference" ■ Updated the following sections: <ul style="list-style-type: none"> Configuring AMM for Additional Database Schemas About Distributed Transactions About the AMM Controller

Version	Date	Description
E16719-17	December 2015	<p>Documentation updates for BRM 7.5 Patch Set 14.</p> <ul style="list-style-type: none"> Added information about the start date parameter to the following sections: <ul style="list-style-type: none"> Syntax for Purging Objects without Removing Partitions Parameters for Purging Objects without Removing Partitions Purging Objects without Removing Partitions Added information about setting the PerfAdvancedTuningSettings business parameter in "Improving Performance by Skipping Previous Total Unpaid Bill for Open Item Accounting Type". Added an entry for PerfAdvancedTuningSettings business parameter in "Billing business_params Entries". Added information about setting the Skip_Prev_Total business profile key in "Improving Performance by Skipping Previous Total for Open Item Accounting Type When Calculating the Current Bill". Added an entry for TimestampRoundingForPurchaseGrant business parameter in "Pricing and Rating business_params Entries". Added the "Logging External User Information in Error Logs" section. Updated the following sections: <ul style="list-style-type: none"> pin_clean_rsvns Using Error Logs to Troubleshoot BRM Billing pin.conf Entries
E16719-18	April 2016	<p>Documentation updates for BRM 7.5 Patch Set 15.</p> <ul style="list-style-type: none"> Added "Publication pin.conf Entries" for the new ece_real_time_sync_db_no CM pin.conf entry. Added the following chapters: <ul style="list-style-type: none"> Using Business Operations Center Business Operations Center Utilities
E16719-19	August 2016	<p>Documentation updates for BRM 7.5 Patch Set 16.</p> <ul style="list-style-type: none"> Added the following section: <ul style="list-style-type: none"> Setting How Long the DM Waits for the Background Startup Process to Finish Updated the following sections: <ul style="list-style-type: none"> General Software Requirements Creating the Data Store in Oracle IMDB Cache Data Manager (DM) pin.conf Entries Tax Calculation pin.conf Entries pin_config_distribution EAI Manager pin.conf Entries Specifying the Number of Retries in a Deadlock

Version	Date	Description
E16719-20	December 2016	Documentation updates for BRM 7.5 Patch Set 17. <ul style="list-style-type: none"> Added information about the following operations to "Using Business Operations Center": <ul style="list-style-type: none"> Synchronizing product catalogs Refunding payments
E16719-21	April 2017	Documentation updates for BRM 7.5 Patch Set 18. <ul style="list-style-type: none"> Added the following sections: <ul style="list-style-type: none"> Improving Performance in Retrieving Product Details During Product Purchase Improving Trial Billing Performance by Enabling General Ledger Collection Updated the following sections: <ul style="list-style-type: none"> Billing business_params Entries Improving Performance by Disabling Unused Features pin_virtual_gen
E16719-22	August 2017	Documentation updates for BRM 7.5 Patch Set 19. <ul style="list-style-type: none"> Updated the following section: <ul style="list-style-type: none"> Subscription business_params Entries
E16719-23	April 2018	Documentation updates for BRM 7.5 Patch Set 21. <ul style="list-style-type: none"> Added the following chapter: <ul style="list-style-type: none"> Backing Up and Restoring Your BRM System
E16719-24	November 2018	Documentation updates for BRM 7.5 Patch Set 22. <ul style="list-style-type: none"> Added the following section: <ul style="list-style-type: none"> Enabling SSL for Customer Center Web Start Deployment Removed the following sections: <ul style="list-style-type: none"> Changing the Interval for Checking New Accounts pin_confirm_logins
E16719-25	December 2019	Documentation updates for BRM 7.5 Patch Set 23. <ul style="list-style-type: none"> Updated the following sections: <ul style="list-style-type: none"> Problems Loading Configuration Objects Subscription business_params Entries

Part I

Basic BRM System Administration

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

- [Starting and Stopping the BRM System](#)
- [Monitoring and Maintaining Your BRM System](#)
- [Using Configuration Files to Connect and Configure Components](#)
- [Implementing System Security](#)
- [BRM OMF Instrumented Objects](#)
- [Configuring Pipeline Manager](#)
- [Controlling Batch Operations](#)
- [About Connection Pooling](#)
- [System Administration Utilities and Scripts](#)
- [SNMP Utilities](#)

Starting and Stopping the BRM System

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

Before you read this chapter, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

About Starting and Stopping BRM Components

You start and stop BRM components by starting and stopping the corresponding *process* for that component.

You can start and stop BRM components by using the following methods:

- The **pin_ctl** utility. See ["Starting a Component by Using the pin_ctl Utility"](#).
- Pipeline Manager **ifw** command.

Choosing the User Name for BRM Components

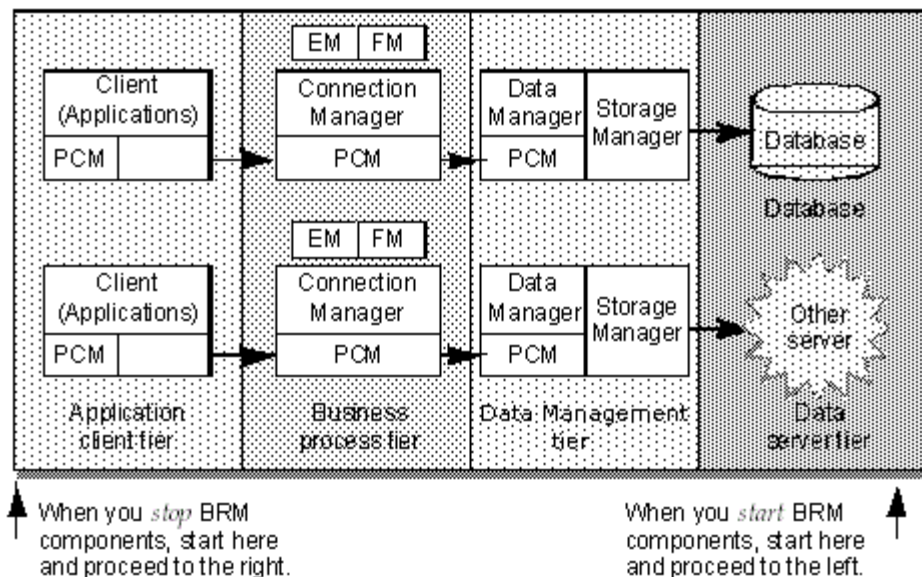
You can start BRM components as user **root**, user **pin**, or any other name you choose. If you start both the BRM database and the BRM components with a user name other than **root**, you have better control over security and administrators that do not have superuser permissions.

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**.

Starting Several BRM Components in Sequence

You must start BRM components in a specific order, starting with the data tier and finishing with the application tier.

The tiers are structured as shown in [Figure 1-1](#):

Figure 1–1 BRM Component Tiers

To start multiple components:

1. Start the BRM database.

The BRM database usually starts automatically when the computer is started. For information on stopping and starting the database, see the documentation for your database server.

2. Start the Data Manager (DM) for your database (**dm_oracle**).

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

3. Start any other DMs, such as these:

- **dm_fusa** for BRM-initiated payment processing

Note: Start the Paymentech credit card simulator before starting DM FUSA. See "Running the Paymentech Simulators" in *BRM Configuring and Collecting Payments*.

- **dm_vertex** for tax calculation

4. Start the daemon or process for any optional features such as these:

- Mailer
- Popper
- Web interface
- Tax

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 optional service integration components.

Tip: Create a custom start script that includes required **pin_ctl start component** commands and **start_component** scripts. Be sure to start components in the proper order.

Stopping Several BRM Components in Sequence

You must stop BRM components in a specific order, starting with the application tier and finishing with the data server tier.

1. Stop the BRM client applications and optional service integration components.
2. If your system uses any of these optional features, stop each daemon or process:
 - Mailer
 - Popper
 - Web interface
 - Tax
3. Stop the CMs.
4. Stop CM proxy if your system uses it.
5. Stop the Connection Manager Master Processes (CMMPs) if your system uses them.
6. Stop all but the database DMs, such as **dm_vertex** for tax calculation
7. Stop the Oracle DM for your database (**dm_oracle**).
8. Close all database sessions, such as SQL*Plus sessions for Oracle.
9. Stop the BRM database.

For information on stopping and starting the database, see the documentation for your Oracle database server.

Starting a Component by Using the pin_ctl Utility

You can use the **pin_ctl** utility to perform the following:

- Start BRM components. See ["Starting BRM Components with pin_ctl"](#).
- Start Pipeline Manager components. See ["Starting Pipeline Manager Components with pin_ctl"](#).
- Retrieve diagnostic data when you start a component. See ["Getting Diagnostic Data When You Start a Component"](#).
- Restart a component. See ["Halting and Restarting a Component with One Command"](#).
- Start a component and clear the log file. See ["Starting a Component and Clearing the Log File"](#).
- Start a base set of BRM components. See ["Starting a Base Set of BRM Components"](#).

Starting BRM Components with pin_ctl

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 a list of valid *component* values, see "[pin_ctl](#)".

For example:

```
pin_ctl start dm_oracle
```

Starting Optional BRM Components with pin_ctl

By default, the **pin_ctl** utility is not configured to start BRM optional components, such as Synchronization Queue Manager DM (**dm_aq**).

To start an optional BRM component by using the **pin_ctl** utility:

1. Add the optional component to the *BRM_home/bin/pin_ctl.conf* configuration file. See "[Customizing pin_ctl.conf for Starting and Stopping Optional Components](#)".
2. Run the **pin_ctl** utility with the **start** action:

```
pin_ctl start component
```

where *component* is the optional component you added in step 1.

For example, to start **dm_aq**:

```
pin_ctl start dm_aq
```

Starting Pipeline Manager Components with pin_ctl

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

1. Edit the *BRM_home/bin/pin_ctl.conf* configuration file. See "[Configuring the pin_ctl Utility](#)".
2. Copy the *Pipeline_home/bin/ifw* binary file to *Pipeline_home/bin/component*, where *component* is the pipeline component name.
3. Go to the *BRM_home/bin* directory.
4. Run the **pin_ctl** utility with the **start** action:

```
pin_ctl start component
```

where *component* is the component you want to start. For a list of valid *component* values, see "[pin_ctl](#)".

Getting Diagnostic Data When You Start a Component

To get diagnostic data about 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**, **-collectdata**, and *component*:

```
pin_ctl start -collectdata component
```


where *component* is the component you want to get data about. For a list of valid *component* values, see "[pin_ctl](#)".

For example:

```
pin_ctl start -collectdata dm_oracle
```

Halting and Restarting a Component with One Command

To halt and restart 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 **restart** action:

```
pin_ctl restart component
```

where *component* is the component you want to halt and restart. For a list of valid *component* values, see "[pin_ctl](#)".

For example:

```
pin_ctl restart dm_oracle
```

Starting a Component and Clearing the Log File

You can clear the log file for a component and then start the component.

Note: If the component is already running, the command just clears the log file and the component continues running.

To start a BRM component and clear the contents of its log file by using the **pin_ctl** utility:

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

```
pin_ctl cstart component
```

where *component* is the component parameter.

For example:

```
pin_ctl cstart dm_oracle
```

Starting a Base Set of BRM Components

You can use the **pin_ctl start all** command to start a customizable set of BRM components. By default, the components are started in this order:

- Oracle Data Manager (DM)
- Email DM
- Connection Manager
- CM Master Process
- Invoice formatter

You can customize the components that the **pin_ctl start all** command starts. You can also create customized "all" commands to start a separate set of components. See "[Creating a Custom "all" Parameter List](#)".

To start a set of BRM components by using the **pin_ctl** utility:

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

```
pin_ctl start all
```

Note: You can use the **-collectedata** parameter to collect diagnostic data about all of the components. See ["Getting Diagnostic Data for a Component by Using the pin_ctl Utility"](#).

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.

Important: 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 ["Starting Several BRM Components in Sequence"](#).

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 master process and one or more child processes. When you start the master 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 BRM Component by Using the pin_ctl Utility

To stop 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 **stop** action:

```
pin_ctl stop component
```

where *component* is the component you want to stop. For a list of valid *component* values, see "[pin_ctl](#)".

For example:

```
pin_ctl stop dm_oracle
```

Getting Diagnostic Data When You Stop a Component

For information about the diagnostic data, see "[Getting Diagnostic Data for a Component by Using the pin_ctl Utility](#)".

To get diagnostic data when you stop a BRM component:

1. Go to the *BRM_home/bin* directory.
2. Run the **pin_ctl** utility with **stop**, **-collectdata**, and *component*:

```
pin_ctl stop -collectdata component
```

where *component* is the component you want to get data about. For a list of valid *component* values, see "[pin_ctl](#)".

For example:

```
pin_ctl stop -collectdata dm_oracle
```

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.

Stopping a Base Set of BRM Components

You can use the **pin_ctl stop all** command to stop a customizable set of components. By default, the components are stopped in this order:

- Invoice formatter
- CM Master Process
- Connection Manager
- Email DM
- Oracle DM

You can customize the components that the **pin_ctl stop all** command stops. See ["Customizing the Components Included in "all""](#) You can also create customized "all" commands to stop a separate set of components. See ["Creating a Custom "all" Parameter List"](#).

To stop a set of BRM components by using the **pin_ctl** utility:

1. Go to the *BRM_home/bin* directory.
2. Run the **pin_ctl** utility with the **stop** action and **all** as the component:

```
pin_ctl stop all
```

Note: You can use the **-collectdata** parameter to collect diagnostic data about all of the components. See ["Getting Diagnostic Data for a Component by Using the pin_ctl Utility"](#).

Starting and Stopping Pipeline Manager Manually

You can stop and start Pipeline Manager by using the command line instead of by using the **pin_ctl** utility.

Important: If you start Pipeline Manager manually, you cannot use the **pin_ctl** utility to control Pipeline Manager (for example, to stop it or to get the status).

Starting Pipeline Manager

You start an instance of Pipeline Manager by using the following command from the *Pipeline_home* directory:

```
bin/ifw -r RegistryFile
```

where *RegistryFile* is the name of the registry file.

Important: If Pipeline Manager cannot establish a connection with the Pipeline Manager database (most likely because the database is down), you receive an error message and the Pipeline Manager startup is canceled.

The general syntax for the **ifw** command and parameters is:

```
ifw -r RegistryFile | -h | -v [-r RegistryFile]
```

where:

-r *RegistryFile*

Starts Pipeline Manager with the specified registry file.

-h

Displays the syntax and parameters.

-v [-r *RegistryFile*]

Displays the version of Pipeline Manager. If you use the **-r** parameter, it also displays the version and name of data and function modules. For example:

```
ifw -v -r conf/wireless.reg
```

```
Module ifw.DataPool.Listener.Module Name: DAT_Listener, Version: 10010
```

Pipeline Manager displays **Ready for processing** when startup procedures are completed.

Stopping Pipeline Manager

Important: If you use HTTP or SNMP monitoring in a Pipeline Manager instance, you must stop all monitoring requests to Pipeline Manager before you stop it. To stop the monitoring requests, stop the master SNMP agent. You can use the kill command, for example:

```
kill -9 master_agent_pid
```

You stop Pipeline Manager by using the following semaphore entry:

```
ifw.Active = FALSE
```

For information on semaphores and how to create semaphore files, see ["Updating Configuration Settings during Runtime by Using Semaphore Files"](#).

Starting and Stopping Individual Pipelines

When you start Pipeline Manager, the Controller starts all pipelines. However, you can stop or restart individual pipelines by using semaphores.

Important: When a pipeline cannot establish a connection with the Pipeline Manager database (most likely because the database is down), you receive an error message and the pipeline startup is canceled.

To start an individual pipeline, use the following semaphore entry:

```
ifw.Pipelines.PipelineName.Active = True
```

where *PipelineName* is the name of the pipeline.

To stop an individual pipeline, use the following semaphore entry:

```
ifw.Pipelines.PipelineName.Active = False
```

where *PipelineName* is the name of the pipeline.

If files are added to the input directory after a pipeline is stopped and before it is restarted, the files are processed in order based on their last modified timestamp. For more information about input processing, see "Configuring EDR Input Processing" in *BRM Configuring Pipeline Rating and Discounting*.

For information on how to create semaphore files, see "[Updating Configuration Settings during Runtime by Using Semaphore Files](#)".

Tip: Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks. See "[Using Perl Scripts to Start and Stop Pipeline Manager](#)".

Restarting Pipeline Manager after an Abnormal Shutdown

Some modules track data in data files. If an abnormal shutdown occurs, you must delete the data files that were in progress when the shut-down occurred.

See the following topics in *BRM Configuring Pipeline Rating and Discounting*:

- Managing the Call Assembling Data Files
- Managing FCT_DuplicateCheck Data Files

Using Perl Scripts to Start and Stop Pipeline Manager

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks.

[Table 1–1](#) describes the files and scripts used for starting and stopping pipelines:

Table 1–1 Scripts for Starting and Stopping Pipelines

Semaphore and Perl script file names	Description
start_all_pipeline.reg start_all_pipeline.pl	<p>Starts all pipelines configured in the start_all_pipeline.reg semaphore file. By default, this list includes these pipelines:</p> <ul style="list-style-type: none"> ■ ALL_RATE ■ PRE_RECYCLE ■ PRE_PROCESS ■ ALL_BCKOUT ■ ALL_RERATE <p>If you add custom pipelines, add their names to the semaphore file according to the default examples.</p> <p>Important:</p> <ul style="list-style-type: none"> ■ Do not run rating pipelines (ALL_RATE, PRE_RECYCLE, and PRE_PROCESS) at the same time that you run the rerating pipeline (ALL_RERATE). Edit the script to specify which pipeline to run. ■ Before running ALL_RERATE, ensure that the backout pipeline (ALL_BCKOUT) has processed all EDRs that were extracted by the Event Extraction tool.
start_all_rate.reg start_all_rate.pl	Starts the ALL_RATE pipeline.
start_DiscountPipeline.reg start_DiscountPipeline.pl	Starts the discount pipeline (DiscountPipeline).
start_main_stop_rerating.reg start_main_stop_rerating.pl	<p>Starts these pipelines:</p> <ul style="list-style-type: none"> ■ PRE_PROCESS ■ PRE_RECYCLE ■ ALL_RATE <p>Stops these pipelines:</p> <ul style="list-style-type: none"> ■ ALL_BCKOUT ■ ALL_RERATE
start_pre_process.reg start_pre_process.pl	Starts the preprocessing pipeline (PRE_PROCESS).
start_pre_recycle.reg start_pre_recycle.pl	Starts the pre-recycle pipeline (PRE_RECYCLE).
start_RealtimePipelineGPRS.reg start_RealtimePipelineGPRS.pl	Starts the real-time GPRS pipeline (RealtimePipelineGPRS).
start_RealtimePipelineGSM.reg start_RealtimePipelineGSM.pl	Starts the real-time GSM pipeline (RealtimePipelineGSM).
start_RealtimePipelineZone.reg start_RealtimePipelineZone.pl	Starts the real-time zoning pipeline (RealtimePipelineZone).
start_recycle.reg start_recycle.pl	Starts recycling rejected CDRs.

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

Semaphore and Perl script file names	Description
start_rerating_stop_main.reg start_rerating_stop_main.pl	Stops these pipelines: <ul style="list-style-type: none"> ■ ALL_RATE ■ PRE_RECYCLE ■ PRE_PROCESS Starts these pipelines: <ul style="list-style-type: none"> ■ ALL_BCKOUT ■ ALL_RERATE
stop_all_pipeline.reg stop_all_pipeline.pl	Stops all the pipelines configured in the stop_all_pipeline.reg semaphore file. By default, this list includes these pipelines: <ul style="list-style-type: none"> ■ ALL_RATE ■ PRE_RECYCLE ■ PRE_PROCESS ■ ALL_BCKOUT ■ ALL_RERATE If you add custom pipelines, add their names to the semaphore file according to the default examples.
stop_all_rate.reg stop_all_rate.pl	Stops the ALL_RATE pipeline.
stop_DiscountPipeline.reg stop_DiscountPipeline.pl	Stops the discount pipeline (DiscountPipeline).
stop_pre_process.reg stop_pre_process.pl	Stops the preprocessing pipeline (PRE_PROCESS).
stop_pre_recycle.reg stop_pre_recycle.pl	Stops the pre-recycle pipeline (PRE_RECYCLE).
stop_RealtimePipelineGPRS.reg stop_RealtimePipelineGPRS.pl	Stops the real-time pipeline for GPRS (RealtimePipelineGPRS).
stop_RealtimePipelineGSM.reg stop_RealtimePipelineGSM.pl	Stops the real-time pipeline for GSM (RealtimePipelineGSM).
stop_RealtimePipelineZone.reg stop_RealtimePipelineZone.pl	Stops the real-time pipeline for zoning (RealtimePipelineZone).

Starting and Stopping Oracle IMDB Cache DM

Before starting or stopping Oracle In-Memory Database (IMDB) Cache DM, ensure that the `TIMESTEN_HOME` environment variable is set to the directory in which you installed the Oracle IMDB Cache database.

Starting IMDB Cache DM

To start Oracle IMDB Cache DM:

1. Go to the `BRM_home/bin` directory.
2. Run the following command:

```
pin_ctl start dm_tt
```


Stopping IMDB Cache DM

To stop Oracle IMDB Cache DM:

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

```
pin_ctl stop dm_tt
```

Monitoring and Maintaining Your BRM System

This chapter provides information and guidelines to help you manage the day-to-day operation of your Oracle Communications Billing and Revenue Management (BRM) system.

About Monitoring BRM

You use the following tools to monitor BRM components:

- The **pin_ctl** utility. Use this utility to start and stop BRM components and to get diagnostic data. See ["Using the pin_ctl Utility to Monitor BRM"](#).
- The **pin_db_alert.pl** utility. Use this utility to monitor key performance indicators (KPIs), which are metrics you use to quantify the health of your database and to alert you to potential risks. See ["Using the pin_db_alert Utility to Monitor Key Performance Indicators"](#).
- Operations Management Framework (OMF) HTTP and SNMP protocols. These protocols provide access to data collected by OMF probes. See:
 - [Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components](#)
 - [Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data](#)

In addition, you can use component-specific diagnostic tools such as:

- Pipeline Manager Diagnostic Data Handler. See ["Using the Diagnostic Data Handler to Get OMF Diagnostic Data"](#).
- Log files. See ["Using Logs to Monitor Components"](#).
- Connection Manager (CM) quality of service (QoS) statistics. See ["Getting Quality of Service Statistics from the CM"](#).
- Operating system commands. See:
 - [Manually Checking the Status of the CM](#)
 - [Manually Checking the Status of the DM](#)

Table 2-1 provides an overview of the system monitoring tools:

Table 2–1 BRM System Monitoring Tools

Monitoring Tool	Functions	Component
pin_ctl utility	Stop and start components. Get diagnostic data. Clear log files.	All system components See "Components Monitored and Controlled by the pin_ctl Utility" .
pin_db_alert.pl utility	Monitor key performance indicators.	Oracle databases
Diagnostic Data Handler	Get application diagnostic data.	Pipeline Manager
Log files	Get status and error messages.	All system components
QoS statistics	Get QoS statistics.	CM
HTTP and SNMP system monitoring	Get instrumentation data from probes. Set configuration values.	Pipeline Manager Real-time pipeline See "BRM OMF Instrumented Objects" .

Components Monitored and Controlled by the pin_ctl Utility

You can use the **pin_ctl** utility to monitor 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
 - Account Synchronization Data Manager
 - Invoice Data Manager
- EAI Java Server
- Invoice Formatter
- Paymentech Answer Simulator
- Pipeline Manager, including:
 - Real-time pipeline
 - Batch pipeline
- Batch Controller
- System Manager
- Node Manager

About Data Collected by OMF

You can use OMF to get instrumentation data from Pipeline Manager. See ["About Operations Management Framework"](#) and ["BRM OMF Instrumented Objects"](#).

You can use two methods to get data:

- SNMP. See ["Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components"](#).
- HTTP. See ["Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data"](#).

Using the pin_ctl Utility to Monitor BRM

You can perform the following monitoring tasks by using the **pin_ctl** utility:

- Use the **status** command to get the current status of the component. See ["Getting the Status of a Component by Using the pin_ctl Utility"](#).
- Use the **clear** command to delete log entries associated with the component (not the file). See ["Clearing Log Files for a Component by Using the pin_ctl Utility"](#).
- Use the **-collectdata** parameter to get diagnostic data when starting, stopping, or checking the status of a component. See ["Getting Diagnostic Data for a Component by Using the pin_ctl Utility"](#).

You also use the **pin_ctl** utility to start, stop, halt, and restart system components. For information about starting and stopping BRM by using the **pin_ctl** utility, see ["Starting and Stopping the BRM System"](#).

For more information, see ["pin_ctl"](#).

Setting Up 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. See ["Configuring the pin_ctl Utility"](#).

Important: (AIX only) To start more than one pipeline process in the same machine, you must assign a different IFW_EVENTHANDLER port for each pipeline process. For example, if you are starting rtp, aaa, and bre in the same AIX machine:

```
rtp env_platform:common env_variable:IFW_EVENTHANDLER_PORT env_val:XXXX1
aaa env_platform:common env_variable:IFW_EVENTHANDLER_PORT env_val:XXXX2
bre env_platform:common env_variable:IFW_EVENTHANDLER_PORT env_val:XXXX3
```

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*

Getting the Status of a Component by Using the pin_ctl Utility

You can get the status of a component at any time.

To get the current status of a component by using the **pin_ctl** utility:

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

```
pin_ctl status component
```

where *component* is the component for which you want the status. For a list of valid component values, see "[pin_ctl](#)".

For example:

```
pin_ctl status dm_oracle
```

You can use the **-collectdata** parameter to get diagnostic data when checking the status of a component:

```
pin_ctl status -collectdata component
```

See "[Getting Diagnostic Data for a Component by Using the pin_ctl Utility](#)".

Clearing Log Files for a Component by Using the pin_ctl Utility

To clear a component's log file by using the **pin_ctl** utility:

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

```
pin_ctl clear component
```

where *component* is the component whose log file you want to clear. For a list of valid component values, see "[pin_ctl](#)".

For example:

```
pin_ctl clear dm_oracle
```

You can also clear log files when you start a component by using the **cstart** command.

See "[Starting a Component and Clearing the Log File](#)".

Getting Diagnostic Data for a Component by Using the pin_ctl Utility

You can use the **pin_ctl** utility to get diagnostic data about a component at the following times:

- Before startup by using the **start** command. The data is collected before the component is started.
- Before shutdown by using the **stop** command. The data is collected after the component is stopped.
- While it is running when you use the **status** command.
- When you run the **restart** and **cstart** commands.

Note: If you collect data during the **stop all** or **status all** commands, data is collected for all components before the command is carried out. For example, if you stop all components, data is collected about all the components, and then they are stopped.

The diagnostic data is written to a file in the component's log directory. The file name is *component.diag.log* (for example, *cm.diag.log*).

When a new file is created, BRM renames the existing file to *component.diag.log.YYYYMMDDhhmmss* (for example, *dm_oracle.diag.log.20060918094046*).

To get diagnostic data about a component by using the **pin_ctl** utility:

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

```
pin_ctl action -collectdata component
```

where:

- *action* specifies the action to be executed (for example, **start**), during which you want to collect diagnostic data.
- *component* is the component for which you want diagnostic data. For a list of valid component values, see "[pin_ctl](#)".

For example:

```
pin_ctl start -collectdata dm_oracle
```

Diagnostic Data Collected by the pin_ctl Utility

- Date and time the data was collected.
- Extended information from the system (for example, system, node, release, and kernel ID).
- Environment variables for the current terminal session.
- System limits for the current terminal session.
- Memory information (for example, available memory).
- Storage device information (for example, available disk space).
- Patch level on the system.
- Kernel parameters.
- Network status showing all sockets, routing table entries, and interfaces.
- Network status summary.
- Inter-process communication facilities status.
- NFS statistics.
- Duration of time that the system has been up.
- All active components.
- All active users.

Configuring the pin_ctl Utility

You can configure the **pin_ctl** utility by editing the **pin_ctl.conf** file. See the following topics:

- [Customizing the Components Included in “all”](#)
- [Customizing pin_ctl.conf for Starting and Stopping Optional Components](#)
- [Creating a Custom “all” Parameter List](#)
- [Customizing the Components List](#)
- [Customizing the pin_ctl Startup Configuration](#)
- [Customizing the pin_ctl Utility Environment Variables](#)
- [Setting the pin_ctl Utility Log Level](#)
- [Configuring the Start and Stop Validation Settings for pin_ctl](#)
- [Customizing snmpset and snmpget Actions](#)
- [Using Custom pin_ctl Configuration Files](#)

The **pin_ctl.conf** file is in *BRM_home/bin*.

Customizing the Components Included in “all”

You can customize the components that are included in the **pin_ctl** **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 cmmmp formatter

all dm_oracle dm_email cm cmmmp 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](#)".

Important: 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.

Customizing pin_ctl.conf for Starting and Stopping Optional Components

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 Synchronization Queue DM (**dm_aq**), you must:

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|:pipeline|->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.
- **:pipeline** identifies the component as pipeline.
- **->dependent_service** specifies one or more components that *service_name* depends 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 “all” Parameter 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:

```
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 cmmmp formatter

all dm_oracle dm_email cm cmmmp 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](#)".

Important: 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. Separate component names by using a space.

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
...
4 rtp:pipeline
4 aaa:pipeline
4 bre:pipeline
4 bre_tt:pipeline
```

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
```

```

1 dm_ifw_sync_1=dm_ifw_sync
1 dm_ifw_sync_2=dm_ifw_sync
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
3 rtp_1=rtp:pipeline
3 rtp_2=rtp:pipeline
3 aaa_1=aaa:pipeline
34 aaa_2=aaa:pipeline

```

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.
# :pipeline should be used if it is Pipeline service [Optional].

```

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

```
start_sequence service_name [=alias_name|:java|:app|:pipeline|->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.
- **:pipeline** identifies the component as pipeline.
- **->dependent_service** specifies one or more components that *service_name* depends 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:

```
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 "[component Parameter](#)".

- *operating_system* can be **hpux_ia64**, **linux**, **aix**, **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. To define environment variables for pipeline registry files, find the following lines:

```
# registry details for pipeline services
aaa env_platform:common env_variable:AAA_REGISTRY env_val:$IFW_
HOME/conf/diameter_charge.reg
rtp env_platform:common env_variable:RTP_REGISTRY env_val:$IFW_
HOME/conf/wirelessRealtime.reg
bre env_platform:common env_variable:BRE_REGISTRY env_val:$IFW_
HOME/conf/wireless.reg
```

5. Add the following line for each pipeline component that uses a registry file:

```
component env_platform:common env_variable:registry_variable env_val:$IFW_
HOME/registry_file
```

where:

- *component* is the pipeline component name. For a list of valid values, see "[component Parameter](#)".
- *registry_variable* is the environment variable to set before starting *component*. The syntax for pipeline registry environment variables is ***_REGISTRY**.
- *registry_file* is the path and file name for the pipeline registry file.

For example:

```
aaa env_platform:common env_variable:AAA_REGISTRY env_val:$IFW_
HOME/conf/diameter_charge.reg
```

6. Save and close the file.

Setting the CTL_SNMP_PATH variable

You can set the CTL_SNMP_PATH variable to one of the following:

- *BRM_home/bin*. For example:

```
common env_platform:common env_variable:CTL_SNMP_PATH env_val:BRM_home/bin
```

- The path of the SNMP third-party software. For example:

```
common env_platform:common env_variable:CTL_SNMP_PATH env_val:/home2/mydir/opt/snmp/bin
```

Setting the pin_ctl Utility Log Level

To set the log level for the **pin_ctl** utility:

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

The syntax is:

```
Control_script_log loglevel:level logfile:log_file
```

where:

- *level* is the log level, which can be:
 - none**: no logging
 - error**: log error messages only (default)
 - warning**: log error messages and warnings
 - debug**: log error messages, warnings, and debugging messages
- *log_file* is the name of the log file.

For example:

```
Control_script_log loglevel:error logfile:pin_ctl.log
```

Note: Instead of always getting debugging information, you can use the **pin_ctl -debug** parameter to get debugging information whenever you run the **pin_ctl** utility. For example:

```
pin_ctl -debug start dm_oracle
```

3. 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:

- 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:
 - For BRM processes, this overrides the *BRM_home* value for the specified component.
 - For pipeline processes, this overrides the *IFW_home* value for the specified component. This is used as the relative path for entries in the registry file. For

example, if a registry entry specifies *./temp*, the pipeline process uses *IFW_home/temp*.

- 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 -- PIN_LOG 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:logpath
```

where:

- *component* is the BRM component. For a list of valid values, see "[component Parameter](#)".
- *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 for BRM processes and the *IFW_home* value for pipeline processes.
- *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.

Customizing snmpset and snmpget Actions

You can run **snmpset** and **snmpget** commands from the **pin_ctl** utility. You can edit the **pin_ctl.conf** file to add **snmpset** and **snmpget** actions.

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

The syntax is:

```
snmp_command servicename probe registry_entry base_OID
```

where:

- *snmp_command* is either **snmpset** or **snmpget**.
- *servicename* is the name of the component. Use the names defined for the **pin_ctl** utility. See ["pin_ctl"](#).
- *probe* is the name of the probe that receives the **snmpset** command. For information about probes, see ["BRM OMF Instrumented Objects"](#).
- *registry_entry* is the registry entry that corresponds to the probe.
- *base_OID* is the base process ID (OID) from the BRM Management Information Base (MIB) file. See ["BRM OMF Instrumented Objects"](#).

3. Save and close the file.

You can now use the **pin_ctl** utility **snmpset** or **snmpget** action by using the probe name. For example:

```
pin_ctl snmpset probe_name component
```

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
```

Using the pin_db_alert Utility to Monitor Key Performance Indicators

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

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 2–2](#) contains a list of supported KPIs and provides the default behavior of their data and validation modules.

Table 2–2 Supported KPIs and Default Behavior

KPI ID	Default Behavior
AuditHistoryAge	<p>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.</p> <p>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.</p> <p>For information on changing the default age thresholds, see "Monitoring the Age of Audit Tables".</p>
AuditTableSize	<p>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.</p> <p>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.</p> <p>For information on changing the default size thresholds, see "Monitoring the Size of Audit Tables".</p>
OldestEventAge	<p>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.</p> <p>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.</p> <p>For information on changing the default event age, see "Monitoring the Age of Events".</p>

Table 2–2 (Cont.) Supported KPIs and Default Behavior

KPI ID	Default Behavior
ACTIVETRIGGERS	<p>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.</p> <p>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.</p> <p>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".</p>
INDEXES	<p>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.</p> <p>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.</p> <p>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".</p>
PROCEDURES	<p>The proceduresList module retrieves a list of stored procedures in the BRM system and writes the stored procedure names and status (VALID or INVALID) to the proceduresList_PROCEDURES.out file. This enables Pipeline Manager to compile data in parallel and to restore it from the precompiled data files file.</p> <p>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.</p> <p>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".</p>

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 2–3](#) are possible for any KPI:

Table 2–3 KPI Status Severity Levels

Severity	Description
CRITICAL	<p>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).</p> <p>Set up alert notifications for critical failures so you can correct such problems immediately and avoid further corruption.</p>
MAJOR	<p>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.</p> <p>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.</p> <p>Set up alert notifications for major failures so you can correct problems immediately and avoid further degradation or data corruption.</p>
MINOR	<p>Performance might be impacted and will need attention in the future. Minor failures involve large audit tables, which might impact pipeline startup time.</p>
WARNING	<p>Performance and functionality both work as expected, but performance may be impacted in the future.</p> <p>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.</p> <p>Warning failures generally do not impact performance and <i>never</i> impact functionality.</p>
NORMAL	<p>No data or performance risks were found.</p> <p>This status is valid only for PASS results.</p>

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"](#).

About Monitoring KPIs

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.

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

For more information, see ["Setting Up KPI Monitoring"](#).

Setting Up KPI Monitoring

The default configuration for monitoring KPIs is defined in the **pin_db_alert.pl** utility's configuration file (*BRM_home/agnostics/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"](#).

Important: 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.

Important: 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:john_smith@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.

Important: 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";
```

For example:

```
DB_USER="brm123";
DB_
PASSWD="&aes|0D5E11BFDD97D2769D9B0DBFBD1BBF7EE03F1642861DFA57502C7FB85A654267";
```

3. Save and close the file.

For more information about encrypting passwords, see "About Encrypting Information" 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 **AuditHistoryAge** 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
```

Note: 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"](#).

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 **SupportInformer75** 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 and Pipeline Manager 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 **pinrev.dat** files.

The information collected from Pipeline Manager includes:

- Configuration files
RDA collects the pipeline configuration data from the **.reg** (registry) and **.dsc** (description) files. For example, RDA collects the configuration data for wireless from the **wireless.reg** and **containerDesc.dsc** files.
- Log files
RDA collects pipeline log data from the process log, pipeline log, and stream log files. For example, RDA collects the log data for wireless from the **processWIRELESS.log** file, the **log_streamRT1.log** file, and so on.
- Other files
RDA collects pipeline installation and version details from the **vpd.properties** and **piperev.dat** files.

To find BRM component information, RDA looks in the following directories:

- *BRM_home*/sys
- *BRM_home*/apps

To find Pipeline Manager information, RDA looks at the registry files.

A complete overview of RDA is provided in the *Remote Diagnostic Agent (RDA) 4 - Getting Started* document. See ["Viewing RDA Documentation"](#).

RDA 4.21 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 4.21.

Installing Remote Diagnostic Agent

RDA is included in the Third-Party package along with Perl and Java Runtime Environment. It automatically gets installed when you install the Third-Party package, in the directory you choose to install the Third-Party software. For more information, see "Installing the Third-Party Software" in *BRM Installation Guide*.

Note:

- RDA is not supported on Windows.
 - RDA collects diagnostic and configuration data for all BRM and Pipeline Manager components and applications *only* from the server on which RDA is running. To collect data for BRM or Pipeline Manager 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 a profile named **SupportInformer75**, 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 **SupportInformer75** profile runs other modules, such as INI, CFG, END, RDSP, and LOAD.

Running Remote Diagnostic Agent

To run RDA:

1. Go to the directory where you installed the Third-Party package and source the **source.me** file:

Bash shell:

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

C shell:

```
source source.me.csh
```

2. To collect BRM system information, verify that the PIN_HOME environment variable is set to the BRM installation directory. By default, it is **/opt/portal**.
3. To collect pipeline log files, verify that the INT_HOME environment variable is set to the Pipeline Manager installation directory. By default, it is **/opt/ifw**.
4. 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
```

5. 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 get online documentation about the BRM module:

```
perl rda.pl -M BRMr7.def
```

- To perform BRM data collection using default values:

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

Important: 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 and Pipeline Manager installations and system configuration. RDA saves all your responses to the **/rda/setup.cfg** file.

Note: You can change the default location of **setup.cfg** file, if required.

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)
>
```

If your database is running on the same server as RDA, RDA prompts for the following database information:

```
S200DB: Controls RDBMS Data Collection

Enter the Oracle SID to be analyzed
Hit 'Return' to accept the default (PortalDB)
>
Enter the location of the spfile or the INIT.ORA (including the directory and file
name)
>
```

To collect BRM and Pipeline Manager system information, RDA prompts for the following BRM information:

```
S380BRM: Collects BRM Software Information

Should RDA collect BRM Software information (Y/N)?
Hit 'Return' to accept the default (Y)
>
Should RDA collect BRM based system information (Y/N)?
Hit 'Return' to accept the default (Y)
>
Enter a pipeline registry file to analyze or . to terminate the list
>
```

Prompts are displayed that apply to other Oracle products. For these cases, choose the default value.

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

Syntax:

```
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 SupportInformer75
```

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

```
perl rda.pl -vdSCRp -p SupportInformer75-DB11g
```

Note: The database version in the **SupportInformer75** profile depends on the version of the database installed for BRM. BRM supports Oracle Database 9i, 10g, and 11g.

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 ["Reporting Problems"](#).

Viewing RDA Documentation

To view the RDA documentation, including the *Getting Started*, *FAQ*, and *Troubleshooting* guides:

1. Go to My Oracle Support (support.oracle.com).
2. In the **Search Knowledge Base** field, enter 330364.1 and click the **Global Search** icon.
3. In the search results, click the **Remote Diagnostic Agent (RDA) - Main Man Page** link.

The **Remote Diagnostic Agent (RDA) - Main Man Page** appears.

4. In the **RDA Main Links** section of the page, click the link for the appropriate guide.

Dumping Business Parameters in XML Format

To dump BRM business parameters (`/config/business_params` objects) in XML format, use the `pin_cfg_bpdump` utility. See "pin_cfg_bpdump" in *BRM Developer's Guide*. For more information about business parameters, see "Using /config/business_params Objects" in *BRM Developer's Guide* and ["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 called **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.

The following example shows output for the subscription business parameter object with three parameters:

```
<RESULTS elem="0">
  <POID>0.0.0.1 /config/business_params 8526 0</POID>
  <CREATED_T>1213082439</CREATED_T>
  <MOD_T>1213082439</MOD_T>
  <READ_ACCESS>G</READ_ACCESS>
  <WRITE_ACCESS>S</WRITE_ACCESS>
  <ACCOUNT_OBJ>0.0.0.1 /account 1 0</ACCOUNT_OBJ>
  <DESCR>Business logic parameters for Subscription</DESCR>
  <HOSTNAME>--</HOSTNAME>
  <NAME>subscription</NAME>
  <PROGRAM_NAME>--</PROGRAM_NAME>
  <VALUE />
  <VERSION />

- <PARAMS elem="0">
  <DESCR>Parameter to enable contract days counter feature.This      needs to be set to 1 if the
accounts contain the resource      contract days counter</DESCR>
  <PARAM_NAME>discount_based_on_contract_days_feature</PARAM_NAME>
  <PARAM_TYPE>1</PARAM_TYPE>
  <PARAM_VALUE>0</PARAM_VALUE>
</PARAMS>

- <PARAMS elem="1">
  <DESCR>Parameter to enable or disable best pricing feature. Enabling this feature will be
effective only if license is loaded for best pricing. 1 means enabled.</DESCR>
  <PARAM_NAME>best_pricing</PARAM_NAME>
  <PARAM_TYPE>1</PARAM_TYPE>
  <PARAM_VALUE>0</PARAM_VALUE>
</PARAMS>

- <PARAMS elem="2">
  <DESCR>Threshold of the number of offerings below which poids of offerings retrieved in
PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS with a database search are cached for use in
subsequent calls to the opcode in the same transaction. If the number of offerings
retrieved is above this threshold, then the use of the cache could become
inefficient.</DESCR>
  <PARAM_NAME>get_offerings_from_cache_threshold</PARAM_NAME>
  <PARAM_TYPE>1</PARAM_TYPE>
  <PARAM_VALUE>100</PARAM_VALUE>
</PARAMS>

- </RESULTS>
```

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.

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"](#).

For information about understanding errors, see ["Using Error Logs to Troubleshoot BRM"](#).

For information about Pipeline Manager log files, see ["About Pipeline Manager Log Files"](#).

Types of Log Files

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

Log Files for System Components

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**).

Log Files for Applications

For BRM applications, log files are named *program.pinlog* (for example, **pin_bill.d.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 storable objects are not recorded in log files.

Log Files for Client Applications

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. For information about the **Infranet.properties** file, see "Setting Global Options" in *BRM Developer's Guide*.

For Payment Tool, the log file **default.pinlog** is located in the same directory as the executable file.

Location of Log Files

The following are the minimum BRM log files:

- **cm.log**
- **cm.pinlog**
- **dm.log**
- **dm_oracle.pinlog**
- **pin_bill.d.pinlog**

Depending on what applications are running, your installation might also have one or more of these log files:

- **dm_email.log**
- **dm_email.pinlog**
- **pin_invoice_gen.log**
- **dm_fusa.log**
- **dm_fusa.pinlog**

Your customizations or special applications might generate their own log files.

You may want to increase the logging level to 2 (see ["Setting the Reporting Level for Logging Messages"](#)) and have your notification script detect and act on warning messages. Log files should be archived weekly to a safe storage area.

Tip: You can write a script to compress the log files and then erase the originals. BRM automatically re-creates new empty log files as required.

Default Log File Locations

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.

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. See ["Locations of Configuration and Properties Files"](#).
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. See ["Starting and Stopping the BRM System"](#).

Tip: You can change the name of the default application's log file by using the `PIN_ERR_SET_LOGFILE` function (see `"PIN_ERR_SET_LOGFILE"` in BRM Developer's Reference).

Note: 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.

Important: To avoid performance degradation, use only level 3 logging for debugging.

To change the severity level for logging:

1. Open the configuration file (`pin.conf` or `.properties`) for the component or application. See ["Locations of Configuration and Properties Files"](#).
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. See ["Starting and Stopping the BRM System"](#).

Important: DMs automatically report errors and warnings and do not use the `loglevel` entry to set reporting level. To show debugging messages for a DM, see ["Increasing the Level of Reporting for a DM"](#).

Getting Debugging Information from Command-Line Utilities

Most BRM utilities use the following command-line parameters:

- `-d`: Set the log level to debug and outputs debug information into the log file. If not set, only error-level information is output. Use this parameter if no errors were reported, but the command was not successful (for example, if a `/config` object was not loaded).
- `-v`: Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter `-v` at the beginning or end of a command to initiate verbose display. To redirect the output to a log file, use the following syntax with the `-v` parameter. Replace *filename.log* with the name of the log file:

command *any_other_parameter* `-v > filename.log`

Dynamically Changing the CM and DM Log Levels

You can dynamically change the log level of the CM and DM without stopping and restarting them.

To change the log levels dynamically:

1. Pass the log level for the CM and the debug flags for the DM in the input flist to the `PCM_OP_INFMGR_SET_LOGLEVEL` opcode.

Note: You change the log levels of the CM and DM at the same time.

You can check the current log levels by calling the `PCM_OP_INFMGR_GET_LOGLEVEL` opcode.

2. Call the opcode by using `testnap`.

All the new processes after this opcode call will use the new CM log levels and DM debug flags.

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. See ["Starting and Stopping the BRM System"](#).

For information on setting the system log level, see ["Setting the Reporting Level for Logging Messages"](#).

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`

Masking Sensitive Data in Log Files

Log files may contain masked fields as configured by your BRM implementation. Subscriber fields, including payment information and user credentials, may be hidden in logs for securing sensitive subscriber data.

See ["About Securing Sensitive Customer Data with Masking"](#) in *BRM Managing Customers* for more information on configuring data masking in system logs.

Maintaining Log Files

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.

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
```

Dealing with Hung and Looping Processes

A *hung* process does not respond in a normal fashion.

A *looping* process uses CPU cycles without doing any useful work.

Checking for Hung Processes

If the CM does not respond to a login attempt, one of the processes in the system might be hung. Check the status of the CM. See ["Monitoring CM 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. See ["Manually Checking the Status of the DM"](#). You can check the database by verifying that it responds to manual SQL commands.

Checking for Looping Processes

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. 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 DM Activity"](#) or ["Monitoring CM 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 ["Getting Help with BRM Problems"](#).)

Monitoring CM Activity

You can check the CM's status and resolve lost TCP connections.

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 master 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 2–4](#) describes the state values:

Table 2–4 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
10	Shutting down the child CM
11	Starting the child CM

Resolving Lost TCP Connections

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.

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.

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 about TCP keepalive options, see the TCP and keepalive documentation for your operating system.

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, Customer Center, 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. See ["Starting and Stopping the BRM System"](#).

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 BRM client applications that use the **Infranet.properties** file:
 1. Open the **Infranet.properties** file in a text editor.

By default, the **Infranet.properties** file is in **C:/Program Files/Common Files/Portal Software**.
 2. Add the following entry to the file:

```
infranet.PcmTimeoutInMsecs= 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.

Note: The timeout value specified in the configuration or **Infranet.properties** file 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 configuration or properties file entry.

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.

Getting Quality of Service Statistics from the CM

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 2-5](#) is collected per opcode:

Table 2–5 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 finish without returning an error.
Timestamp of minimum latency	The timestamp when the minimum latency occurred.
Maximum latency	The slowest elapsed time that this opcode took to finish 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:

```
- 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:

```
- 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. See ["Starting and Stopping the BRM System"](#).

Monitoring DM Activity

You can check the status of a Data Manager (DM) 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.

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.

- [Checking the DM Status in flist Format](#)
- [Checking the DM Status in a Report Format](#)

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.

BRM displays the status of the DM in flist format.

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

[Table 2–6](#) describes the fields in */status_dm*.

Table 2–6 */status_dm* Object Fields

<i>/status_dm</i> 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. <ul style="list-style-type: none"> ■ DMSHM_ALIVE 0x00001000 ■ DMSHM_DYING 0x00002000 ■ DMSHM_DEAD 0x00004000 ■ DMSHM_INITING 0x00008000 ■ DMSHM_RESTARTING 0x00010000

Table 2–6 (Cont.) /status_dm Object Fields

/status_dm Object Field	Description
PIN_FLD_TATTLE_TALE	<p>This flag is reset each time you retrieve the DM status report. This enables you to see what happened since the last DM report.</p> <ul style="list-style-type: none"> ■ 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 front end.
PIN_FLD_FLAGS	<p>Specifies the internal state of the DM front end. These states are used for the internal working of the Oracle DM.</p> <ul style="list-style-type: none"> ■ DMSHM_ALIVE 0x00001000 ■ DMSHM_DYING 0x00002000 ■ DMSHM_DEAD 0x00004000 ■ DMSHM_INITING 0x00008000 ■ DMSHM_RESTARTING 0x00010000
PIN_FLD_TATTLE_TALE	<p>This flag is reset each time you retrieve the DM status report. This enables you to see what happened since the last DM report.</p> <ul style="list-style-type: none"> ■ 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	<p>Specifies the internal state for a DM context:</p> <ul style="list-style-type: none"> ■ 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

Table 2–6 (Cont.) /status_dm Object Fields

/status_dm Object Field	Description
PIN_FLD_DM_FE_STATE	Specifies the current front-end state in the DM context. <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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 executed. Note: To find an opcode's number, see the opcode header files in the <i>BRM_home/include/ops</i> 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 a Report Format

To check the status of the DM in a report format:

1. Find the process ID (PID) of the master DM 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 master DM 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. Heap High Water Mark (HWM) is 80% of the total allocated heap blocks of memory. When the **#used** crosses HWM, DM would throw an error. 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.

The maximum number of bytes used out of DM shared memory (**dm_bigszie**) is indicated by **big_max_used**. The maximum number of heap blocks used is indicated by **hblock_max_used**. For example, let us say the heap blocks used are 3000. If 1000 blocks are released, the number of used blocks are 2000. In this case **hblock_max_used** will be 3000, which is the largest value of the heap blocks used.

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 2-7](#) shows the flag bit values:

Table 2-7 DM Flag Bit Values

Value	Flag	Description
0x8	IO output	Never cleared for back ends.
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 2-8](#), the back-end state values are a bit mask flag:

Table 2-8 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 2–9](#) describes the front-end state values:

Table 2–9 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. There are three settings to control which debugging information is logged:

- **DM_DEBUG** variables control the logging of opcode-processing debug messages.
- **DM_DEBUG2** variables control the logging of data dictionary processing debug messages.
- **DM_DEBUG3** variables debug the SQL statements produced by different parts of the DM.

The *BRM_home/include/dm_debug.h* file contains definitions of the flags you can set. You specify which individual flags you want to enable for each setting by summing the values of the flags and including the sum in an environment variable or in the DM configuration (**pin.conf**) file.

For example, to log information about transaction tracing, you set **DM_DEBUG** to **0x70**, which is the sum of the following individual 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 three settings (**DM_DEBUG**, **DM_DEBUG2**, and **DM_DEBUG3**).

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 statements in the DM's configuration (**pin.conf**) file. You specify each setting separately as in the following example:

```
- dm dm_debug      0xFFFF003FF
- dm dm_debug2     0x10
- dm dm_debug3     0x10
```

See ["Editing the Configuration File to Set Debug Options"](#).

- For **dm_oracle** and **dm_tax**, you must specify the debugging information as environment variables. You set a separate environment variable for each debug setting.

See ["Using Environment Variables to Set Debug Options"](#).

You can dynamically change the debugging level without stopping the DM. For more information, see ["Dynamically Changing the CM and DM Log Levels"](#).

Using Environment Variables to Set Debug Options

To set debug 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 the environment variable for debugging. 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.

Editing the Configuration File to Set Debug 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. See ["Locations of Configuration and Properties Files"](#).
3. Edit the three debugging 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 debugging entries in the configuration file. 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.

You can also dynamically set or change this variable when the DM is running. See ["Dynamically Changing the CM and DM Log Levels"](#).

Replacing Failed DM Child Processes

All DMs, such as IMDB Cache DM, Paymentech DM and Email DM are set to automatically replace child processes that have stopped. This feature prevents the system from losing DM processes because 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. See ["Locations of Configuration and Properties Files"](#).
2. Change the value of the **dm_restart_children** entry to 0.
3. Save and close the file.
4. Stop and restart the DM. See ["Starting and Stopping the BRM System"](#).

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 ["Dealing with Hung and Looping Processes"](#).

Monitoring Pipeline Manager

For information about improving Pipeline Manager performance, see ["Optimizing Pipeline Manager Performance"](#).

Monitoring Pipeline Manager Memory Usage

You can use the `MemoryMonitor` module to monitor Pipeline Manager memory during startup and while it is processing files. You set a threshold for the amount or percentage of memory that determines when Pipeline Manager should issue a warning or gracefully shut down. You can set the thresholds as a percentage or as kilobytes or megabytes.

For example, if you set **ShutdownFreeMemLimit** to 50 and **ScaleUnit** to **M**, Pipeline Manager shuts down gracefully when the remaining free system memory reaches 50 MB. If you set **WarningFreeMemLimit** to 10 and **ScaleUnit** to **P**, Pipeline Manager logs a warning when the remaining free system memory reaches 10 percent.

See "Memory Monitor" in *BRM Configuring Pipeline Rating and Discounting*.

Monitoring Pipeline Manager EDR Throughput

You can monitor the following statistics for each pipeline:

- Number of event data records (EDRs) since startup.
- Accumulated EDR processing time since startup.
- Total number of EDRs since startup, independent of any transaction. This number is incremented after every processed EDR.
- Total number of EDRs after the transaction ended. This number is not incremented until the current transaction has ended.
- The real-time EDR count increments after each EDR is processed, while the transaction count increments EDR count only after transaction/file processing is ended.
- Number of transactions since startup.
- EDRs per second (throughput). This data includes the timestamp of when the measurement was taken.

You can use the Operations Management Framework (OMF) HTTP and SNMP protocols to access the data. See ["Pipeline Statistics Probes"](#).

Getting Recent Pipeline Log File Entries

You can display recent log file entries in the OMF HTTP server. See ["Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data"](#). The entries are also included in the Diagnostic Data Handler output file. See ["Using the Diagnostic Data Handler to Get OMF Diagnostic Data"](#).

The log messages are stored in a circular buffer that stores the last 1000 log messages. See ["Log File Probes"](#).

You can change the number of error messages stored in the buffer. To do so, edit the **CircularBufferSize** registry entry in the **ITO** section.

For example:

```
ProcessLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
```

```

        LogLevel          = Debug
...
    CircularBufferSize    = 100
}

```

Monitoring IMDB Cache DM

In addition to the **pin_ctl** utility, you can use the following features to monitor the IMDB Cache DM:

- Core dumps. See ["Generating the IMDB Cache DM Core Dump"](#).
- Log files. See ["Troubleshooting IMDB Cache DM errors"](#).
- Opcode latency statistics. See ["Getting Opcode Statistics from IMDB Cache DM"](#).
- System tables. See ["About the Global Transaction System Tables and Views"](#).

Generating the IMDB Cache DM Core Dump

To generate the IMDB Cache DM core dump:

1. Go to the system where IMDB Cache DM is started.
2. Enter the following command:

```
setenv sbUtDumpCore 1
```

Troubleshooting IMDB Cache DM errors

By default, IMDB Cache DM reports errors and warnings in the **dm_tt.pinlog** file. Additionally, you can use environment variables to set debug options to report debugging information in the log file.

IMDB Cache DM logs system activities in the **/var/portal/7.5/dm_tt/dm_tt.pinlog** file. Any error in the IMDB Cache DM is reported in the **dm_tt.pinlog** file, and the error number is returned to the Connection Manager (CM). You should monitor this log file daily to detect and diagnose system problems. You might want to create a script file to periodically scan the log file and notify you if it detects any error messages.

Getting Opcode Statistics from IMDB Cache DM

You can collect statistics about opcode performance from IMDB Cache DM. IMDB Cache DM prints the opcode stack with details about the total time spent at Oracle IMDB Cache and at the BRM database. This data can be used to compare opcode performance and for debugging purposes. For example, if the database operation is taking more time, check the database statistics to ensure the database is running optimally.

To get opcode statistics from IMDB Cache DM, set the following entry in the IMDB Cache DM and Oracle DM **pin.conf** files:

```
- dm enable_pcm_op_call_stack 1
```

The opcode stack is printed for the whole transaction after the transaction is committed or aborted.

The following is a sample opcode stack output:

```

0.000000000 Enter PCM_OP_TRANS_OPEN (0x0)

0.000187000 Exit  PCM_OP_TRANS_OPEN (0x0) TT Time    0.000000000

```

```

42.727118000 Enter PCM_OP_READ_FLDS (0x0)

42.729580000 Exit  PCM_OP_READ_FLDS (0x0) TT Time      0.001472000

566.135870000 Enter PCM_OP_TRANS_ABORT (0x0)

566.136405000 Exit  PCM_OP_TRANS_ABORT (0x0) TT Time      0.000187000

```

To customize the opcode stack size, set the following optional configuration entry in the IMDB Cache DM and Oracle DM **pin.conf** files.

```
- dm max_pcm_op_call_stack_entries Size
```

About the Global Transaction System Tables and Views

You can find information about global transactions in the following system tables and views:

- GLOBAL_TRANS_T. See ["About the GLOBAL_TRANS_T Table"](#).
- GLOBAL_PENDING_TRANS_T. See ["About the GLOBAL_PENDING_TRANS_T Table"](#).
- DBA_2PC_PENDING. See ["About the DBA_2PC_PENDING View"](#).

About the GLOBAL_TRANS_T Table

The GLOBAL_TRANS_T table stores the status of each **Active** global transaction in the system. The **pin_tt_schema_gen** utility creates an instance of this table for each logical partition in your system.

[Table 2–10](#) describes the columns in the GLOBAL_TRANS_T table. This table contains one row for each back end in its associated logical partition.

Table 2–10 GLOBAL_TRANS_T Table Description

Column Name	Description
BACKEND_ID	Specifies the back-end number. This table contains a row for each back end supported by the logical partition. The number of back ends (or rows) depends on the dm_n_be parameter in the IMDB Cache DM pin.conf file. For more information, see "Configuring DM Front Ends and Back Ends" .
GLOBAL_TRANS_NAME	The global transaction name used by each back end. Each back end assigns the same name to each global transaction that it opens. The global transaction name uses the following naming convention: T_LogicalPartNum_BackendID where: <i>LogicalPartNum</i> is the logical partition ID, such as 0.1.0.1 or 0.0.0.2. <i>BackendID</i> is the back-end number. The Oracle database converts this name into hexadecimal format. For example, T_0.2.0.1_3 is converted into 0.545F302E322E302E315F33. Important: The global transaction name is shared between the back ends of an active-standby pair.
GLOBAL_TRANS_STATE	Specifies the state of the global transaction: <ul style="list-style-type: none"> ■ 0 specifies that the global transaction is in reset state. ■ 1 specifies that the global transaction is in precommit state.

About the GLOBAL_PENDING_TRANS_T Table

The GLOBAL_PENDING_TRANS_T table stores the status of each **Pending** global transaction in the system.

The IMDB Cache Manager installer creates this table in each schema in your BRM database. You must be logged in as the database schema user to access this table.

[Table 2–11](#) describes the columns in the GLOBAL_PENDING_TRANS_T table.

Table 2–11 GLOBAL_PENDING_TRANS_T Table Description

Table Column	Description
DATABASE_NO	Specifies the database number assigned to the logical partition. For example, 0.1.0.1.
BACKEND_ID	Specifies the back-end number. This table contains a row for each back end supported by the logical partition. The number of back ends (or rows) depends on the dm_n_be parameter in the IMDB Cache DM pin.conf file. For more information, see " Configuring DM Front Ends and Back Ends ".
GLOBAL_TRANS_NAME	The global transaction name used by each back end. Each back end assigns the same name to each global transaction that it opens. The global transaction name uses the following naming convention: T _{LogicalPartNum_BackendID} where: <i>LogicalPartNum</i> is the logical partition ID, such as 0.1.0.1 or 0.0.0.2. <i>BackendID</i> is the back-end number. The Oracle database converts this name into hexadecimal format. For example, T_0.2.0.1_3 is converted into 0.545F302E322E302E315F33. Important: The global transaction name is shared between the back ends of an active-standby pair.
LOCAL_TRANS_NAME	Specifies the local transaction identifier, in the following the format: <i>Integer.Integer.Integer</i> For example: 13.45.844769
COMMIT_ADVICE	Recommends whether the Oracle Database Administrator should commit or rollback the transaction. This is set to either Commit or Rollback .
FAIL_DATE	Specifies the data and timestamp when the global transaction failed.

About the DBA_2PC_PENDING View

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 the IMDB Cache DM and Oracle DM processes to access this view, your system administrator must grant read privileges to the BRM database user.

[Table 2–12](#) describes the columns in the DBA_2PC_PENDING view.

Table 2–12 DBA_2PC_PENDING View Description

Column Name	Description
LOCAL_TRAN_ID	<p>For extended architecture (XA) transactions:</p> <p>Specifies the branch qualifier element of an XA transaction ID (XID), which uniquely identifies the local BRM database branch of the XA transaction.</p> <p>For IMDB Cache Manager global transactions:</p> <p>Specifies the local transaction identifier in the format <i>integer.integer.integer</i>. For example: 13.45.844769. When the connection's LOCAL_TRAN_ID and GLOBAL_TRAN_ID are the same, the node is the transaction's global coordinator.</p>
GLOBAL_TRAN_ID	<p>For XA transactions:</p> <p>Specifies the global transaction ID element of an XID, which uniquely identifies the XA transaction.</p> <p>For IMDB Cache Manager global transactions:</p> <p>Specifies the global database identifier in the format <i>global_db_name.db_hex_id.local_tran_id</i>, where <i>db_hex_id</i> is an eight-character hexadecimal value used to uniquely identify the database. This common transaction ID is the same on every node for a distributed transaction. When the connection's LOCAL_TRAN_ID and GLOBAL_TRAN_ID are the same, the node is the transaction's global coordinator.</p>
STATE	<p>Specifies the transaction's state, which can be one of the following values:</p> <ul style="list-style-type: none"> ■ 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.
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.

Maintaining IMDB Cache DM

This section describes steps you can take to maintain the IMDB Cache DM, including:

- [Handling Active IMDB Cache DM Failure](#)
- [Handling Active Node Failure](#)
- [Managing Cache Group Data](#)

Handling Active IMDB Cache DM Failure

In a high-availability system, when an active IMDB Cache DM fails, its associated data store is not notified of the failure, so the data store's status remains active. This prevents the standby data store from becoming active.

Because its associated data store is still on standby, the standby DM rejects all CM requests with the `PIN_ERR_NOT_ACTIVE` error to indicate that it is in standby mode and not accepting requests. (The `PIN_ERR_NOT_ACTIVE` error is recorded as `PIN_ERR_NOT_PRIMARY` in the CM log file.)

Therefore, if an internal IMDB Cache DM error prevents Oracle Clusterware from restarting a DM, you must manually change the standby data store's state to active. This enables the standby DM to switch its state to active and process the requests redirected to it by the CM.

All CM requests will fail until either the active or standby IMDB Cache DM establishes a connection with an active data store.

Handling Active Node Failure

When Oracle IMDB Cache goes down, you must restore the data store by detaching and reattaching the data store to the grid and then re-creating the schema and reloading the BRM objects.

To restore the data store, do the following:

1. Detach and reattach the data store to the grid. See *Oracle In-Memory Database Cache User's Guide* for information on how to detach and reattach data store to a grid.
2. Using `ttIsql`, run `tt_schema.sql` on Oracle IMDB Cache to re-create the BRM cache groups schema.
3. Using `ttIsql`, run the load SQL file on Oracle IMDB Cache to reload the BRM objects into the cache groups.

Note: In a high-availability system, Oracle Clusterware handles detaching and reattaching the data store to the grid. See ["How IMDB Cache DMs Fail Over"](#).

Managing Cache Group Data

Data in the cache groups is stored in shared memory. To avoid running out of shared memory, purge expired BRM objects from the Oracle IMDB Cache to free shared-memory space. Additionally, you can configure an aging policy for the cache groups to purge least-recently-used (LRU) objects.

See ["About Managing Data in Oracle IMDB Cache"](#) for more information about purging BRM objects from the Oracle IMDB Cache.

Finding and Fixing Global Transaction Errors

Your Oracle database administrator should check the `GLOBAL_PENDING_TRANS_T` table periodically for **Pending** global transactions.

Perform the following for each new record in the `GLOBAL_PENDING_TRANS_T` table:

1. In the `GLOBAL_PENDING_TRANS_T` table, check the new record's ["commit_advice"](#) and ["local_trans_name"](#) column values.

- Determine the record's commit state in the DBA_2PC_PENDING view. To do so, connect to the BRM database, log in as the SYSDBA user, and enter the following SQL command:

```
select GLOBAL_TRAN_ID, STATE from DBA_2PC_PENDING where LOCAL_TRAN_ID =
'LocalTranID';
```

where *LocalTranID* is the record's local transaction ID as specified in the "local_trans_name" column of the GLOBAL_PENDING_TRANS_T table.

The query returns the record's commit state. If the state is set to **Prepared**, proceed to the next step.

- Fix the error by forcing either a roll back of the transaction or a commit.
 - If the "commit_advice" column from step 1 was set to **Rollback**, connect to the BRM database, log in as the SYSDBA user, and enter the following SQL command:

```
rollback force 'LocalTranID';
```

- If the "commit_advice" column from Step 1 was set to **Commit**, connect to the BRM database, log in as the SYSDBA user, and enter the following SQL command:

```
commit force 'LocalTranID';
```

For example, assume the GLOBAL_PENDING_TRANS_T table includes the entries shown in [Table 2–13](#).

Table 2–13 Sample GLOBAL_PENDING_TRANS_T Table Entries

database_no	backend_id	global_trans_name	local_trans_name	commit_advice	fail_date
0.1.0.2	1	0.545F302E322E302E315F33	13.45.844769	rollback	01-MAR-11 10.45.53.417743 AM
0.1.0.1	2	0.545F302E322E302E315F33	42.12.550822	commit	01-FEB-11 10.45.53.417743 AM

To fix the **Pending** global transaction in the first row:

- Note that the record's "local_trans_name" is set to **13.45.844769** and the "commit_advice" is set to **Rollback**.
- Determine the record's commit state in the DBA_2PC_PENDING view. Log in to the BRM database as the SYSDBA user and execute the following SQL command for local transaction ID 13.45.844769:

```
select GLOBAL_TRAN_ID, STATE from DBA_2PC_PENDING where LOCAL_TRAN_ID =
'13.45.844769';
```

Rollback

- Force a rollback of the transaction, because the returned commit state was **Rollback**:

```
rollback force '13.45.844769';
```


Monitoring Customer Center Activity

You can configure Customer Center to send list information to a log file by using the SDK to modify the **Infranet.properties** file. You can use this information to monitor Customer Center activity and to resolve problems.

See "Using Customer Center SDK" in *BRM Developer's Guide*.

Monitoring Hardware and Operating Systems

To monitor your system using standard tools, use monitoring utilities such as **vmstat**, **sar**, and **top** on UNIX, or use OS performance monitors such as Glance on HP-UX IA64 systems.

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 Version Numbers of Components

You can check the version numbers of all BRM and pipeline components installed on a machine by using the **pinrev** and **piperev** utilities. These utilities return the following information for each component, ServicePak, FeaturePak, and patch installed on a machine:

- Product name
- Version number
- Components
- Build time
- Installation time

Tip: Run these utilities 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 *BRM_home/bin* directory and run the **pinrev** utility:

pinrev

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

```
PRODUCT_NAME=Portal_Base
VERSION=7.5.0
COMPONENTS= "Portal Base Install Scripts","Rogue Wave Libraries","Common Database
Scripts","batch_controller","pin_billd","pin_cfg_bpdump","pin_inv","cmmp","cm_
proxy","sample","dm_email","pin_export_price","config_export","credit_
control","sox_unlock_service","adu","infmgr","infmgr_cli","dm_invoice","DM
Feature","dm_oracle","bip_udf","formatter","null","nmgr","pin_
subscription","testnap","uei","cm","Java PCM Files","Shared Objects for
Communication","Shared Perl Modules","Common Install Scripts",
BUILD_TIME= 5-23-2011 14:3:7
INSTALLED_TIME=Mon, 23 May 2011 15:27:45 -0700
```

Tip: To print a report of the version information, direct the output of the **pinrev** utility to a file with a **.csv** extension, which you can open in Microsoft Excel. For example:

pinrev > BRM.csv

Important: The **pinrev** utility does not display information about the uninstallation of any BRM component. Only installation information is displayed.

Checking Pipeline Component Version Numbers

To check which pipeline components are installed on your machine, go to the *Pipeline_home/tools* directory and run the **piperev** utility:

piperev

The pipeline displays the versions of all products installed on your machine. For example:

```
PRODUCT_NAME=Pipeline
VERSION=7.5.0
COMPONENTS= "Common files","Pipeline Framework Files","Pipeline Sample
Files","Pipeline TimesTen(TT) Multi Partition","Pipeline Database
Scripts","Pipeline Misc Files","Pipeline Tools files","PDK files","Rogue Wave
Files",
BUILD_TIME= 9-28-2011 13:10:51
INSTALLED_TIME=Thu, 29 Sep 2011 17:33:27 -0700
```

Tip: To print a report of the version information, direct the output of the **piperev** utility to a file with a **.csv** extension, which you can open in Microsoft Excel. For example:

piperev > BRM.csv

Important: The **piperev** utility does not display information about the uninstallation of any pipeline component. Only installation information is displayed.

Using the Diagnostic Data Handler to Get OMF Diagnostic Data

Use the Diagnostic Data Handler to collect analysis data during a crash, exception, or critical error or by performing a snapshot. You can use the Diagnostic Data Handler with Pipeline Manager.

When the Diagnostic Data Handler collects data, it creates a text file that includes information obtained from instrumentation probes. The information includes:

- The stack trace of the Pipeline Manager process.
- Log file entries from the pipeline log circular buffers. See ["Getting Recent Pipeline Log File Entries"](#).
- Diagnostic and performance data collected by Operations Management Framework (OMF) probes. The relevant data includes:

- HTTP server
- SNMP
- Real-time pipeline statistics
- EDR statistics

See "[BRM OMF Instrumented Objects](#)".

You specify the diagnostic file name by editing the **DiagnosticFileName** registry entry. If an existing diagnostic file exists, it is renamed to include the process ID and the date and time that the new file was created. For example:

diagnostic.dat.3418.20060824_113734

You can manually remove old files as needed. You should archive the data files regularly.

By default, diagnostic files are stored in *Pipeline_home/log*.

The **DiagnosticDataHandler** entry is a top-level section in the registry:

```
DiagnosticDataHandler
{
    DiagnosticFilePath = ./log
    DiagnosticFileName = diagnostic.dat
}
```

For more information, see "Diagnostic Data Handler" in *BRM Configuring Pipeline Rating and Discounting*.

Getting a Snapshot from the Diagnostic Data Handler

You can get a snapshot by using the **snmpset** command to set the **createSnapshot** probe value to **True**. When you get a snapshot, a diagnostic file is created.

About Operations Management Framework

Operations Management Framework (OMF) provides a framework for monitoring and controlling the BRM system. OMF is implemented for the following components:

- Pipeline Manager, including batch pipeline and real-time pipeline.
- Multi-threaded framework (MTF).

OMF includes the following components:

- **Instrumentation Probe API.** This component includes *probes* that gather system data and can control processes.

A probe is part of the code for a component. It collects and sets instrumentation data about that component. For example:

- A reference object cache (ROC) synchronization probe can collect information about precommits, postcommits, and rollbacks.
- A Diagnostic Data Handler probe can create a snapshot of the Pipeline Manager system.

- **Instrumentation Probe Broker.** This component provides data from the Instrumentation Probe API to the instrumentation protocols. The Instrumentation Probe Broker runs as part of a Pipeline Manager instance.

- **Instrumentation Protocol Plugins.** These components provide an interface for client tools and web browsers. There are two protocols:
 - **SNMP.** This protocol uses the SNMP daemon to provide instrumentation data to client tools, and to diagnostic tools such as the Pipeline Manager Diagnostic Data Handler.

You can use SNMP utilities to get and set instrumentation data. To do so, you find the instrumentation object IDs (OIDs) in the MIB.

For more information, see ["Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components"](#).

- **HTTP.** This protocol uses a web interface to provide instrumentation data to web applications. BRM includes a default XML style sheet that you can customize to display selected information.

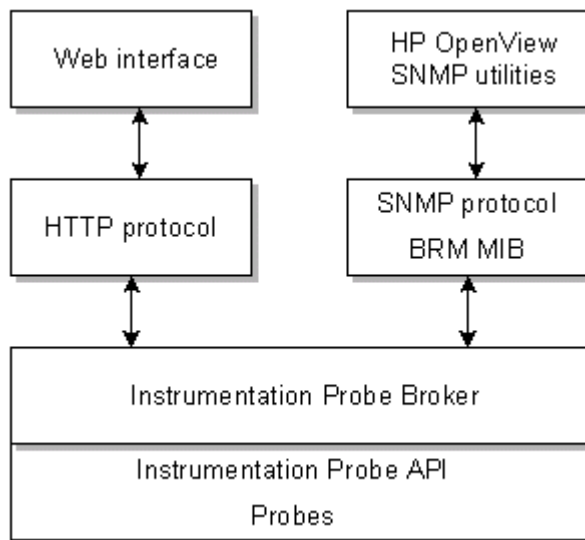
See ["Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data"](#).

Note: The Instrumentation Protocol Plugins have their own probes (for example, to record the number of SNMP GET operations).

Typically, SNMP is used for monitoring the system, and the HTTP interface is for more interactive use.

Figure 2–1 shows the OMF components:

Figure 2–1 OMF Components



About Probe Data

Probes can handle data in the following ways:

- **Attribute probe.** This is the simplest form of data, consisting of a name/value pair. For example:

Number of Startup Threads: 20

- **Group probe.** This is a list of name/value pairs about a related process. For example:

```
Thread Info:
Pool Type      Round Robin
Thread Count   10
```

- **Table probe.** This is a list of groups. For example:

```
DMO Server Configuration:
Name           Host Name           Port Number
DMO Server 1   dmo1.corp.com       13093
DMO Server 2   dmo2.corp.com       13093
```

- **BigTable probe.** This returns large amounts of data (for example, the contents of a log file).

BigTable probes are not supported by SNMP. Therefore, you can only display data from them by using the HTTP protocol.

Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components

SNMP (Simple Network Management Protocol) is a widely-used protocol for monitoring network equipment, computer equipment, and other devices.

When SNMP is configured, you can use SNMP utilities to get and set instrumented data. BRM includes the AGENT++ SNMP utilities. See ["SNMP Utilities"](#).

Important

You can use SNMP utilities other than AGENT++ (for example, NetSNMP). If you use the AGENT++ SNMP utilities, you cannot use symbolic SNMP names in SNMP commands. For example, instead of using the following command:

```
snmpWalk 10.196.129.31 portal.components.mtf.connectionConfigurations.dmoTable.dmoEntry -P20761 -S
```

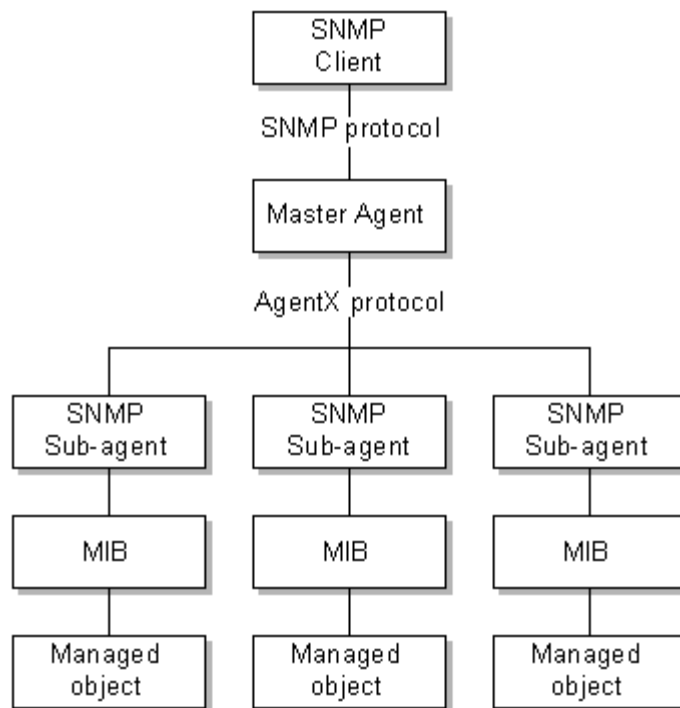
Use the following command:

```
snmpWalk 10.196.129.31 1.3.6.1.4.1.3512.1.5.2.2.1 -P20761 -S
```

About the SNMP Components

The SNMP architecture consists of one master agent per machine, and multiple subagents. Each subagent reads a dynamic MIB file to support changes to the objects being monitored. The subagents use the AgentX protocol to communicate with the master agent. The SNMP clients use the SNMP protocol to communicate with the master agent.

[Figure 2–2](#) shows the SNMP components.

Figure 2-2 SNMP Components

Installing SNMP

Install the BRM SNMP package before installing the BRM Third-Party Software package. The SNMP package includes the following:

- Agent++ SNMP server and configuration files.
- Agent++ SNMP client tools.
- The base BRM MIB file.

Configuring SNMP Components

The SNMP instrumentation protocol runs as an SNMP subagent. SNMP subagents can read from and write to instrumented objects defined in the MIB. subagents connect to the master agent by using the AgentX subagent protocol. For information on configuring and running SNMP subagents, see the SNMP documentation.

To start the SNMP master agent, use the following command:

```
master_agent -l Master_agent_port -x AgentX_port &
```

where *AgentX_port* must be the same as defined in the registry file.

For more information, see the SNMP documentation.

Problems Using SNMP on Oracle Solaris 10

If you are unable to run SNMP utilities, complete the following steps to solve the problem:

1. Open the **etc/system** file using a text editor.
2. Add the following entries:

```
set ip:do_tcp_fusion=0x0
set ip:tcp_fusion_rcv_unread_min=0
```

3. Save and close the file.
4. Restart the system.

Enabling SNMP Instrumentation Data Collection

To enable or disable instrumentation data collection, use the **Instrumentation** section in the Pipeline Manager registry file. If you enable these entries, the SNMP subagent starts when Pipeline Manager starts.

Important: You can enable instrumentation data collection in any Pipeline Manager instance, including those for rating and rerating.

The **Instrumentation** section includes the following entries:

- Use the **ProbeBroker** section to point to the directory that contains probe information files. The default is *Pipeline_home/instrumentation*.
- Use the **SnmpServer** entry to configure the SNMP protocol.
- Use the **Port** entry to define the SNMP AgentX port number.

Important: The port must be the port you assigned when you configured the SNMP master agent and subagent.

- Use the **WaitTimeout** entry to define how long to wait, in milliseconds, before reconnecting to the master agent and reinitializing the MIB.
The default is 10 milliseconds.
- Use the **ProcessDescription** entry to provide a name for the process being monitored. You see the name when you run SNMP commands, for example, and SNMP walk. Providing different names is helpful when you run more than one of the same type of process on a single host.

The default is:

```
user:process_name:registry_file
```

where:

- *user* is the name of the user who ran the process.
- *process_name* is the name of the process.
- *registry_file* is the name of the registry file that configured the process.

The following is a sample **Instrumentation** section:

```
Instrumentation
{
  ProbeBroker
  {
    ProbeInfoFilePath = ./instrumentation
  }
  SnmpServer
  {
```

```
Port = 11960
ProcessDescription = ifw
WaitTimeout = 10
}
HttpServer
{
Port = 12019
StyleSheetFile = ./instrumentation/portal_omf.xsl
PortalLogoFile = ./instrumentation/portal_omf.gif
}
}
```

Important: If you use SNMP monitoring in a Pipeline Manager instance, stop all monitoring requests to Pipeline Manager before you stop it. To stop the monitoring requests, stop the master SNMP agent. You can use the **kill** command. For example:

```
kill -9 master_agent_pid
```

About the BRM MIB

The BRM MIB defines the structure of the managed data in the BRM system. It includes all the processes and probes that are registered as instrumented objects.

The MIB is described in the **PORTAL-MIB.txt** file in *BRM_home/instrumentation*. For a description of the MIB, see ["BRM OMF Instrumented Objects"](#).

About Dynamic Object IDs

You can run multiple instances of the same type of process on a single host. Therefore, these separate processes must be identified by SNMP. To do so, BRM creates object IDs (OIDs) dynamically by using:

- The base OID from the MIB.
- A process ID from a process table.
- A registry ID from a registry table.
- For table probes only, an instance ID.

The OID for a probe uses the following format:

```
1.3.6.1.4.1.3512.1.component_id.module_id.1.probe_id.process_id.registry_id.instance_id
```

To use SNMP to access the probe value, you must find the process ID, registry ID, and instance ID. For example, to find the OID for a specific process **batchSizeLimit** entry:

1. See ["BRM OMF Instrumented Objects"](#) to find the name of the probe.
2. To find the base OID, look in the MIB file or in ["BRM OMF Instrumented Objects"](#). For example:

```
1.3.6.1.4.1.3512.1.2.1.1.1.1
```

3. Use the **snmpWalk** command on the Process table to find the process ID for the component you want to find the value for.

The OID for an entry in the process table is:

```
1.3.6.1.4.1.3512.1.101.1
```

The **snmpWalk** command is:


```
snmpWalk host_name 1.3.6.1.4.1.3512.1.101.1 -Pport -S
```

For example:

```
snmpWalk frisco 1.3.6.1.4.1.3512.1.101.1 -P44293 -S
```

The results show:

```
1.3.6.1.4.1.3512.1.101.1.1.1 = 1
```

The process ID is **1**.

4. Use the **snmpWalk** command on the registry table to find the registry ID.

The OID for an entry in the registry table is:

```
1.3.6.1.4.1.3512.1.102.1
```

So the **snmpWalk** command is:

```
snmpWalk host_name 1.3.6.1.4.1.3512.1.102.1 -Pport -S
```

5. Find the instance ID.

For all probes that are not in a probe table, the instance ID is **0**. In this case, the MIB file shows that **batchSizeEntry** is a group probe:

```
transactionManagerGroupEntry ::= SEQUENCE {
    batchSizeLimit          Integer32,
    loggingOff              DisplayString,
    transactionManagerGroupIndex Integer32
}
```

The instance ID is **0**.

The OID for this instance of **batchSizeEntry** is:

```
1.3.6.1.4.1.3512.1.2.1.1.1.1.1.18.0
```

You can use that OID to get or change the value. For example, to get the value:

```
snmpGet frisco 1.3.6.1.4.1.3512.1.2.1.1.1.1.1.18.0 -P44293
```

Tip: To get all available probe values in the process, you can run the **snmpWalk** command at the top level of the MIB structure. For example:

```
snmpWalk frisco 1.3.6.1.4.1.3512.1 -P44293 -S
```

For more information about using probe OIDs to get data, see ["About Probe Data"](#) and ["Getting and Setting Instrumentation by Using SNMP"](#).

About Instance IDs for Table Probes

The instance ID for a table probe is the row number in the table. For example, the **numSnmpGetRequests** probe is part of a table that shows SNMP requests as shown in [Table 2-14](#):

Table 2–14 Instance IDs for Table Probes

SNMP MIB Table Index	MIB Table Name	SNMP MIB Table OID	Number of 'GET' Requests	Number of 'GETNEXT' Requests	Number of 'SET' Requests
1	ProcessTable	1.3.6.1.4.1.3512.1.101.1		2	
2	RegistryTable	1.3.6.1.4.1.3512.1.102.1		98	

An `snmpWalk` command on the `numSnmGetRequests` probe gives these results:

```

1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.1 = 2
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.2 = 98
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.3 = 6
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.4 = 20
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.5 = 8
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.6 = 1
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.7 = 6
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.8 = 6
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.9 = 6
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.10 = 14
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.11 = 12
1.3.6.1.4.1.3512.1.4.2.2.1.1.6.5.12 = 5

```

The last number in the OID is the instance ID, which corresponds with the table row, 1-4. The value shown in the **Number of GET Requests** column is the probe value.

About the Process Table

The process table uses these OIDs:

```

processTable (101)
    processEntry (1)
        processIndex (1)
        processDescr (2)

```

- **processIndex** is the process ID assigned in the table.
- **processDescr** is the description of the process (for example, `ifw`). This description is defined in the **Instrumentation** section of the registry file. See ["Enabling SNMP Instrumentation Data Collection"](#).

For example, an SNMP walk could give these results, where the process ID is **1** and the process description is **ifw**:

```

1.3.6.1.4.1.3512.1.101.1.1.1 = 1
1.3.6.1.4.1.3512.1.101.1.2.1 = ifw

```

Note: The last number in the OID is the row number, which is the same as the process ID.

Each process can have multiple registry settings, each of which needs its unique ID. Therefore, these registered objects are identified in the registry table.

About the Registry Table

The registry table uses these OIDs:

```

registryTable (102)
    registryEntry (1)

```

```
registryIndex (1)
registryName (2)
```

- **registryIndex** is the ID assigned in the table.
- **registryName** is the name used in the registry file (for example, **ifw** or **ifw.Pipelines.ALL_RATE**).

For example, an SNMP walk could give these results:

```
1.3.6.1.4.1.3512.1.102.1.1.1.2 = 2
1.3.6.1.4.1.3512.1.102.1.2.1.2 = ifw.SignalHandler
```

Note: The last number in the OID is the row number, which is the same as the ID.

Getting and Setting Instrumentation by Using SNMP

To get and set instrumentation data, use the SNMP tools installed in *BRM_home/bin*. The following SNMP utilities are included:

- [snmpBulk](#)
- [snmpDiscover](#)
- [snmpGet](#)
- [snmpNext](#)
- [snmpNextAsync](#)
- [snmpPasswd](#)
- [snmpSet](#)
- [snmpWalk](#)
- [snmpWalkThreads](#)

Sample SNMP Input and Output

This section presents sample input and output for the **snmpGet**, **snmpSet**, and **snmpWalk** commands.

snmpGet

In the following sample, the master agent is on balrog/28093:

```
$ snmpGet sampleserver 1.3.6.1.4.1.3512.1.1.1.1.1.1.36.3.0 -P12345
SNMP++ Get to sampleserver SNMPV1 Retries=1 Timeout=1000ms Community=public
Oid = 1.3.6.1.4.1.3512.1.1.1.1.1.1.36.3.0
Value = Startup complete
```

snmpSet

In the following sample, the master agent is on sampleserver/1234.:

```
snmpSet sampleserver 1.3.6.1.4.1.3512.1.1.6.1.1.1.32.7.0 -P12345
SNMP++ Set to sampleserver SNMPV1 Retries=1 Timeout=1000ms SET-community=public
GET-community=public
Oid = 1.3.6.1.4.1.3512.1.1.6.1.1.1.32.7.0
Current Value = << WRITE-ONLY PROBE >>
Value Type is Octet String
Please enter new value: yes
```

```
Set Status = Success
```

```
MTF / Version R2 10092 stopped at 24.08.2007 13:35:04
```

snmpWalk

In the following sample, the master agent is on sampleserver/12345:

```
snmpWalk sampleserver 1.3.6.1.4.1.3512.1.1.1.1.1 -P12345 -S
SNMP++ snmpWalk to kabini2 SNMPV1 Retries=1 Timeout=1000ms Community=public
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.35.0 = true
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.2.0 = Normal
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.3.0 = Startup complete
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.4.0 = true
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.5.0 = 10
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.6.0 = 5
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.7.0 = 5
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.8.0 = 0
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.9.0 = 20
1.3.6.1.4.1.3512.1.1.1.1.1.1.1.1.36.10.0 = DEFAULT, REALTIME, DEFAULT_INACTIVE, REALTIME_
INACTIVE
End of SUBTREE Reached
Total # of Requests = 11
Total # of Objects = 10
```

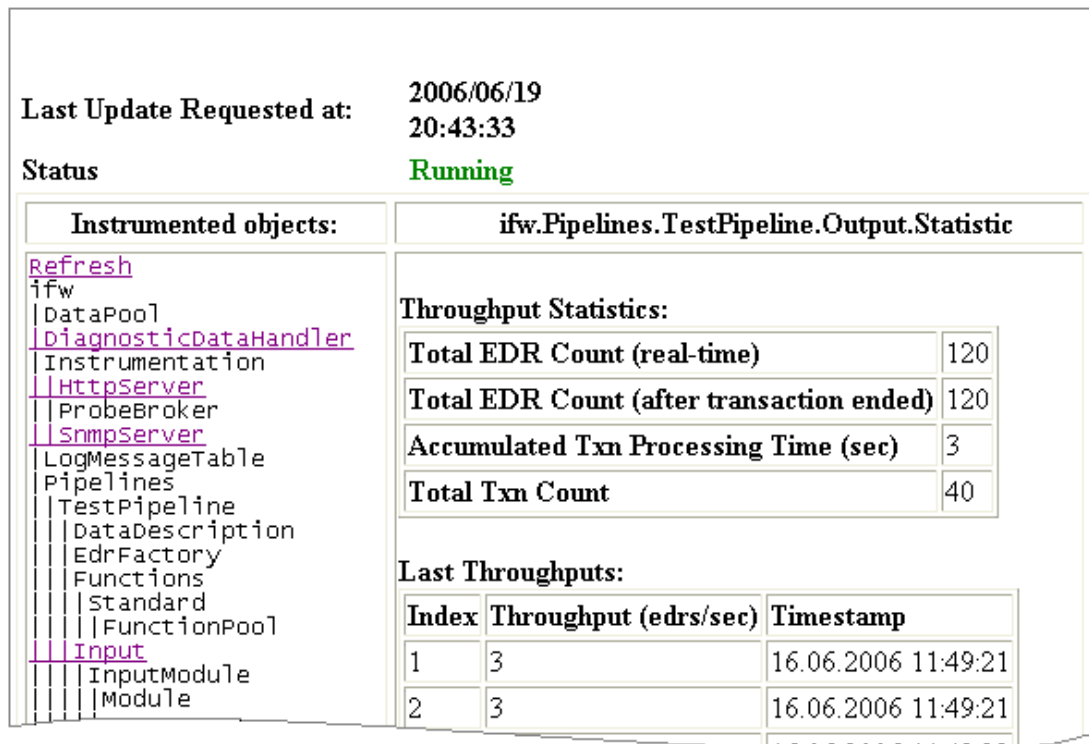
Using the HTTP Instrumentation Protocol to Read OMF Instrumentation Data

To get instrumentation data from the HTTP server, you configure the HTTP server in the Pipeline Manager registry files. You then send the URL for the data you want to retrieve.

Note: You can read instrumentation data from the HTTP server, but you cannot set it.

Figure 2–3 shows the HTTP display:

Figure 2–3 HTTP Instrumentation Display



Enabling HTTP Display of Instrumentation Data

To enable or disable instrumentation data collection, use the **Instrumentation** section in the Pipeline Manager registry files. If you enable these entries, the HTTP server starts in a thread when Pipeline Manager starts.

Important: You can enable instrumentation data collection in any Pipeline Manager instance, including those for running real-time rating and rerating.

The **Instrumentation** section includes the following entries:

- Use the **ProbeBroker** section to point to the directory that contains probe information files. The default is *Pipeline_home/instrumentation*.
- Use the **HttpServer** section to configure the HTTP protocol.
 - Use the **Port** entry to specify the port number for the HTTP server.
 - Use the **StyleSheetFile** and **PortalLogoFile** entries to specify the path to the XML style sheet and the logo displayed in the web interface.

The following is a sample **Instrumentation** section:

```
Instrumentation
{
  ProbeBroker
  {
    ProbeInfoFilePath = ./instrumentation
  }
  SnmpServer
```

```
{
  Port = 11960>
  ProcessDescription = ifw
}
HttpServer
{
  Port = 12019
  StyleSheetFile = ./instrumentation/portal_omf.xsl
  PortalLogoFile = ./instrumentation/portal_omf.gif
}
}
```

Important: If you use HTTP monitoring in a Pipeline Manager instance, stop all monitoring requests to Pipeline Manager before you stop it. To stop the monitoring requests, stop the master SNMP agent. You can use the **kill** command. For example:

```
kill -9 master_agent_pid
```

Displaying Instrumentation Data in a Web Browser

To display instrumentation data in a web browser, use a URL with this format:

http://*host_name:port/registry_entry*

- The host name and port number are those configured in the registry file.
- The registry entry is an entry from the Pipeline Manager registry files.

For example, to get all instrumentation from Pipeline Manager, use **ifw**, the top-level registry entry:

http://*host_name:port/ifw*

Customizing the Data Displayed by the HTTP Instrumentation Protocol

BRM includes a default style sheet to display HTTP instrumentation data. You can customize the style sheet and logo to display selected instrumentation data.

The default style sheet is **portal_omf.xsl**. It is installed in *BRM_home/instrumentation*.

The default logo is **portal_omf.gif**. It is installed in *BRM_home/instrumentation*.

Backing Up and Restoring Your BRM System

This chapter describes the tasks that you perform to back up and restore your Oracle Communications Billing and Revenue Management (BRM) system.

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"](#).

Important: 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:

Important: 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:
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 vpd.properties vpd.properties_75
```

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

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

```
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.

Important: 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.

Using Configuration Files to Connect and Configure Components

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) configuration and properties files, including where to find them and how to edit them.

For information about a specific entry in a configuration file, look directly in the file. Each entry includes a description and guidelines for changing the values.

For information about the optimal values for tuning your BRM system, see ["Improving BRM Performance"](#).

About Configuration and Properties Files

The primary purpose of configuration and properties files is to enable the different components of BRM to communicate with each other (see ["About Connecting BRM Components"](#)). The configuration and properties 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**).
- BRM programs based on Perl scripts read configuration and properties information from a file named **pin.conf.pl**.

You can use any text editor to edit configuration files.

Important: 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 Entry Syntax

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.

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 following platform-dependent extensions:

- **.so** on HP-UX IA64, Oracle Linux, and Oracle Solaris
- **.a** on AIX

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**. For example, you can use other tags to define separate sets of FMs for multiple CMs on the same computer.

Configuration entries for the base-rating FM resemble the following example:

```
- cm fm_module ../../lib/fm_rate_pol.so fm_rate_pol_config - pin
```

The entry shows a policy FM that must always be included with its corresponding base FM.

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

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 Environment Variables" in *BRM Installation Guide*.

[Table 4–1](#) shows the environment variables that can be referenced in BRM configuration files (**pin.conf**):

Table 4–1 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
```

Locations of Configuration and Properties Files

Each daemon can have its own configuration file, or two or more can share a configuration file.

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_bill/pin.conf*).
- In the application folder (for example, *BRM_home/Application_home/Infranet.properties*).

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.

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.

Common Properties File Entry Syntax

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
```

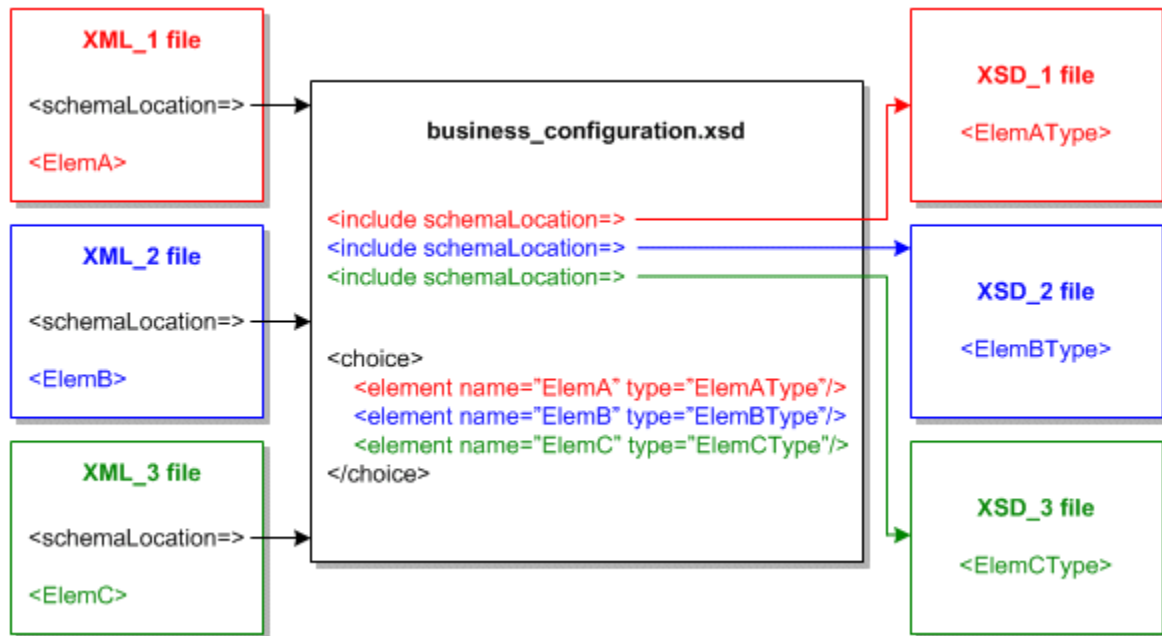
About Validating XML Configuration Files

After you edit the contents of an XML configuration file, a load utility typically validates the contents against the file's schema definition. If the contents do not conform to the schema definition, either the utility returns an error or the load operation fails.

XML files are not directly linked to their schema definition files. Instead, they are linked to one of the following XSD *reference* files:

- **BRM_home/apps/pin_billd/business_configuration.xsd**
- **BRM_home/sys/data/config/business_configuration.xsd**

The XSD reference file associates multiple XML files with their schema definition file. Each XML file contains a schema location pointer to the reference file, and the reference file contains a pointer to the XML file's schema definition. [Figure 4-1](#) shows an example.

Figure 4–1 XML File Validation by *business_configuration.xsd*

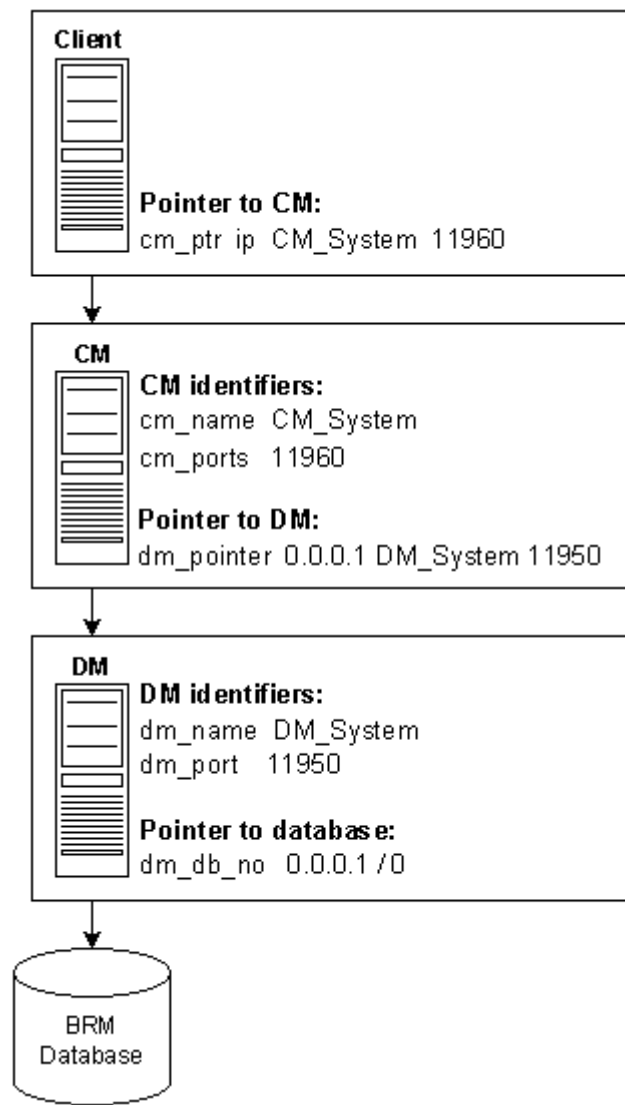
To validate the contents of XML_1, XML_2, or XML_3 in [Figure 4–1](#), **business_configuration.xsd** is called rather than XSD_1, XSD_2, or XSD_3.

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. See ["Reconfiguring a BRM Component"](#).

[Figure 4–2](#) shows how entries in configuration files link components.

Figure 4-2 BRM Component Connections

In [Figure 4-2](#), 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 4-2](#), 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

Important: 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 4–2 Database and Port Number Entries

Program	Database Number	Port Number
The database DM (dm_oracle)	0.0.0.1	12950
The second database DM for a multischema system (dm_oracle)	0.0.0.2	12951
Each additional dm_oracle for multischema systems	0.0.0. <i>n</i>	12950 + (<i>n</i> – 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. <i>n</i> (<i>n</i> > 2)	12813+
Email notification manager: dm_email	0.0.2.1	12821
Each additional dm_email	0.0.2. <i>n</i>	12820 + <i>n</i>
Activity Log Data Manager of a BRM satellite: dm_opstore	0.0.4.1	12841
Each additional dm_opstore	0.0.4. <i>n</i>	12840 + <i>n</i>
LDAP: dm_ldap	0.0.5.1	12851
Each additional dm_ldap	0.0.5. <i>n</i>	12850 + <i>n</i>
Wireless Provisioning DM, dm_prov_telco	0.0.10.2	10970
The Connection Manager (CM) or Connection Manager Master Process (CMMP)	NA	11960
Each additional CM or CMMP	NA	11960 + <i>n</i>
System Manager (inmgr)	NA	11981
Node Manager (nmgr)	NA	11991
The Account Synchronization DM, dm_acctsync	0.0.9.9	11999
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. For example, set the **scoped** and **assign_account_obj** options when you enable branding. You use the **dm_attributes** entry with these options:

- **scoped**: Enables scoping enforcement. This option is used only when you use branding. It restricts database access for separate brands. You can disable this option if branding is not used. See "Data Scoping" in *BRM Developer's Guide*.
- **assign_account_obj**: Assigns an owning account reference when the object is created. When you use branding, all objects are associated with an account. You can disable this option if branding is not used. See "Changing the Owner of a Discount Sharing Group" in *BRM Managing Accounts Receivable*.
- **searchable**: Restricts access to certain database schemas in multischema environments, and indicates that the DM is a database DM. See "Running the pin_multidb.pl Script on the Primary Installation Machine" in *BRM Installation Guide*.

This example specifies a DM pointer in the CM **pin.conf** file:

```
- cm    dm_pointer      0.0.0.1 ip 156.151.2.168      33950 # dm_oracle
- cm    dm_attributes   0.0.0.1 scoped,assign_account_obj,searchable
- cm    dm_pointer      0.0.0.2 ip 156.151.2.168      34950 # dm_oracle
- cm    dm_attributes   0.0.0.2 scoped,assign_account_obj,searchable
```

Connecting a Data Manager to the BRM Database

Use the following DM **pin.conf** file entries in [Table 4–3](#) to connect a DM to the BRM database. These entries are used by multiple DMs, such as DM Oracle and the Account Synchronization DM.

Table 4–3 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 you installed BRM, 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 you installed BRM, but you can change it.

Creating Configuration Files for 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 4–4](#) shows the common utility **pin.conf** entries:

Table 4–4 Common Utility *pin.conf* Entries

Entry	Description
cm_ports	<p>Specifies a pointer to the CM or CMMP.</p> <p>Use a separate entry for each CM or CMMP.</p> <p>Each entry includes three values:</p> <ul style="list-style-type: none"> ■ <i>protocol</i>: ip ■ <i>host_name</i>: the name or IP address of the computer running the CM or CMMP ■ <i>port</i>: the port number of the CM or CMMP on this computer <p>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.</p> <p>See "About Connecting BRM Components".</p>
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.
userid	<p>Specifies the database number and service type for the BRM database.</p> <p>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.</p> <p>The database number, in the form 0.0.0.<i>n</i>, is the number assigned to your BRM database when you installed the system. The default is 0.0.0.1.</p>

In addition, the **pin.conf** entries in [Table 4–5](#) are used by multithreaded application (MTA) utilities:

Table 4–5 MTA Utilities *pin.conf* Entries

Entry	Description
children	<p>Specifies the number of worker threads spawned to perform the specified work. The default is 5.</p> <p>Important: This entry is mandatory.</p> <p>For more information, see the following sections:</p> <ul style="list-style-type: none"> ■ Tuning the Number of Children for Billing Utilities ■ Setting the Number of Children for Invoice Utilities ■ Controlling Thread Load on Your Multithreaded Application
fetch_size	<p>Specifies the number of objects received from the database in a block and cached in system memory for processing. The default is 5000.</p> <p>Important: This entry is mandatory.</p> <p>See the following sections:</p> <ul style="list-style-type: none"> ■ 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.

Table 4–5 (Cont.) MTA Utilities *pin.conf* Entries

Entry	Description
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. <ul style="list-style-type: none"> 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 iterative step search for account information in the BRM database continues in a loop when the number of objects returned by a search step is 0: <ul style="list-style-type: none"> 0: stop the iterative step search. 1: continue the iterative step search. Use the loop_forever entry if the application times out before a search is complete. 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.
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 <i>BRM Developer's Guide</i> .
per_batch	Specifies the number of objects processed by each worker thread in batch mode. The default is 500. Important: This entry is mandatory. See "Tuning the Batch Size for Billing Utilities (per_batch)" .
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 <i>BRM Developer's Guide</i> .
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.

Table 4–5 (Cont.) MTA Utilities *pin.conf* Entries

Entry	Description
return_worker_error	Specifies whether to return an error code when any thread encounters an error: <ul style="list-style-type: none"> ■ 0 specifies to <i>not</i> return an error code. ■ 1 specifies to return an error code. The default is 0.
sleep_interval	Specifies the time in seconds before the iterative step search for account information in the BRM database is resumed within a loop. The default is 60.

Reconfiguring a BRM Component

Each BRM component—CMs, DMs, and applications—gets configuration information from its configuration file, which the component reads when it starts. Changes you make to a configuration file take effect the next time you start the program.

Important: Most configuration file entries require that you restart the CM, but some do not. For information about restart requirements, see ["Business Logic *pin.conf* Reference"](#) and ["System Administration *pin.conf* Reference"](#).

To reconfigure a BRM component:

1. Edit and save the configuration file (***pin.conf***) for the component. See ["Locations of Configuration and Properties Files"](#) and ["About Configuration and Properties Files"](#).
2. If required, stop and restart the component. See ["Starting and Stopping the BRM System"](#).

Running Non-MTA Utilities in Multischema Systems

Utilities built on the MTA framework can perform global searches across all database schemas in a multischema system when configured to do so. See ["Using Multithreaded Applications with Multiple Database Schemas"](#) in *BRM Developer's Guide*.

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

- **pin_clean** (see "pin_clean" in *BRM Configuring and Collecting Payments*)
- **pin_deposit** (see "pin_deposit" in *BRM Configuring and Collecting Payments*)
- **pin_ledger_report** (see "pin_ledger_report" in *BRM Collecting General Ledger Data*)
- **pin_mass_refund** (see "pin_mass_refund" in *BRM Managing Accounts Receivable*)
- **pin_recover** (see "pin_recover" in *BRM Configuring and Collecting Payments*)
- **pin_refund** (see "pin_refund" in *BRM Managing Accounts Receivable*)
- [pin_config_distribution](#)

To run non-MTA utilities in your 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_bildd/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_bildd* 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_bildd* 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_bildd* directory and run the utility.
5. Repeat steps 3 and 4 for each remaining secondary schema in your system.

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 "Using **/config/business_params** Objects" in *BRM Developer's Guide*.

Retrieving /config/business_params Objects

When you retrieve a **/config/business_params** object with **pin_bus_params**, you use the **-r** parameter. To retrieve the object, use the following command:

```
pin_bus_params [-h] -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 [-h] 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.

Implementing System Security

This chapter explains how to set up permissions and passwords for your Oracle Communications Billing and Revenue Management (BRM) system.

Before you read this chapter, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

For information on authenticating and authorizing customers, see "Managing Customer Authentication" in *BRM Managing Customers*.

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*.

Basic Security Considerations

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 the "Critical Patch Updates and Security Alerts" article on the Oracle Technology web site:

<http://www.oracle.com/technetwork/topics/security/alerts-086861.html>

Understanding the BRM Environment

When planning your BRM implementation, consider the following:

- **Which resources must be protected?**

- You must protect customer data, such as credit-card numbers.
- You must protect internal data, such as proprietary source code.
- You must protect system components from being disabled by external attacks or intentional system overloads.
- **Who are you protecting data from?**

For example, you must 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 the Oracle Technology Network web site:

<http://docs.oracle.com>

Storage of Passwords in Configuration Files

By default, the BRM installer stores database and account passwords in encrypted form in the **Infranet.properties** and **pin.conf** configuration files. The BRM applications can automatically decrypt the passwords when retrieving them from the configuration files.

By default, the passwords in the configuration files are encrypted in the Advanced Encryption Standard (AES) format. For more information, see "About AES Encryption" in *BRM Developer's Guide*.

Note: To encrypt passwords for client applications or optional managers that are not part of base BRM, or are associated with customizations, use the **pin_crypt_app** utility. For details, see "About Encrypting Passwords" and "Encrypting Passwords Manually" in *BRM Developer's Guide*.

Passwords for the Connection Manager (CM) 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

Configuring Access to the BRM Database

The Data Manager (DM) configuration file (*BRM_home/sys/dm_oracle/pin.conf*; where *BRM_home* is the directory in which BRM is installed) specifies the user name and password for logging in to the BRM database. This is the user name and password you specified when you installed and configured the database. By default, the user name is **pin** and the password is **pin**.

Note: You can automatically encrypt passwords for all BRM components (including **dm_oracle**) simultaneously by running the **encryptpassword.pl** script. For more information, see "About the encryptpassword.pl Script" in *BRM Developer's Guide*.

Configuring 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

With that information, the CM queries the database. If the information is authenticated, the CM grants the application access to the database.

When you install BRM, the system creates a single user account, called **root** and identified by the password **password**, with general permission to the BRM system. This 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.

The default login name and password provided with the system are **root.0.0.0.n** (where *n* is your database number) and **password**.

Caution: You must change the password after installing BRM to prevent unauthorized access to BRM.

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.

You also must 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 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*) in a text editor.
2. Change the **cm_login_module** entry from **cm_login_pw001.dll** to **cm_login_null.dll**:

```
- cm_login_module cm_login_null.dll
```
3. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).
4. 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**.

Tip: 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"](#).

Configuring the Maximum Number of Invalid Login Attempts

To configure the maximum number of invalid login attempts:

1. Run the following command:

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

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

```
<MaxLoginAttempts>5</MaxLoginAttempts>
```
3. Set the value (the default is 5) to the maximum number of login attempts.
4. Save the file as **bus_params_act.xml**.
5. Run the following command, which loads the change into the **/config/business_params** object:

```
pin_bus_params bus_params_act.xml
```

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

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
7. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

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.
- BRM optional service integration components, such as RADIUS Manager, provide all login information from the configuration file for that application.

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.

Changing Default Passwords for Java-Based Client Applications

The BRM Java-based client applications' passwords are located in their **Infranet.properties** files as encrypted text. To change and encrypt new passwords for an application, log in to the server hosting the application and use the **pin_crypt_app** utility to encrypt the new password manually. For details, see ["About Encrypting Passwords"](#) and ["Encrypting Passwords Manually"](#) in *BRM Developer's Guide*.

Login Information for Payment Tool

To change the default login information for Payment Tool:

1. Open the **C:\Windows\PaymentTool.ini** file.
2. Edit the login entries.
3. Save and close the file.

Login Information for Optional Service Integration Components

To change the default login information for optional service integration components:

1. Open the configuration file for the application, which you can find in the directory for that application.
2. Follow the guidelines in the configuration file to provide information for these configuration entries:
 - **userid**
 - **login_type**
 - **login_name**
 - **login_pw**

Creating a Customer Service Representative Account

Set up accounts for your company's CSRs so they can create and manage customer accounts.

Before creating CSR accounts, create and load a CSR plan, which defines the services available to CSRs. You do this in Pricing Center. See "Creating CSR Plans" in *BRM Setting Up Pricing and Rating*.

You can create CSR accounts in either Customer Center or Permissioning Center. After a CSR account exists, you can make changes to it only in Customer Center.

To create CSR accounts:

- **In Customer Center:** See "Creating a CSR Account" in the Customer Center Help.
- **In Permissioning Center:** In People view, choose **Actions - Create New Person**. For more information, see the Permissioning Center Help. For information on password settings for CSR accounts, see "[Managing CSR Passwords](#)".

Changing the Login Name and Password for an Account

To change the login information for an account, use Customer Center. You can log in to the account you want to change, or you can log in to the BRM **root** account or another account with permission to change the password. You can change the login name and password for each service associated with the account. See the Customer Center Help for more information.

Important: The BRM **root** account comes with the default password of **password**. To prevent unauthorized access to your system, change this password for both services associated with the **root** account.

Logging Customer Service Representative Activities

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.

Note: To log CSR activities, session-event logging must be turned on. Session-event logging is on when the CM `pin.conf` file's `login_audit` entry is set to **1** or commented out. 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` storable class. 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` storable class.

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 *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
- Product cancellation
- Product modification
- Product purchase
- Refunds
- Service login name change
- Settlements
- Top-ups
- Write-offs

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. For information on the types of permissions that you can set, see ["Permission Types"](#).

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 the 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 resources that apply to products 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 can set up permissions using both Permissioning Center and Customer Center. However, you must have proper permissions to add, change, or delete permissions. Usually, only a person with **root** access, such as a system administrator, is granted permission to change CSR permissions. For more information, see ["About Managing Permissions in Permissioning Center"](#).

Note: If your company uses a proprietary application for administering accounts, you can write your own code to set and enforce permissions.

You can also restrict access to accounts in a specific brand. To do this, use Access Privileges in Configuration Center. See "About Granting Access to Brands" in *BRM Managing Customers*.

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. For more information, see ["About Managing Roles"](#).
- **Manage permissions.** You can add and delete permissions. For more information, see ["About Managing Permissions in Permissioning Center"](#).
- **Manage CSRs.** You can create CSR accounts, assign CSRs to roles, and unassign CSRs from roles.

Note: You can create CSRs by using Customer Center or Permissioning Center. You assign individual permissions by using Customer Center, and you assign role-based permissions to CSRs by using Permissioning Center.

You can view the permissions included in a role and the CSRs assigned to a role by using the Role view and People view in Permissioning Center.

Using Permissioning Center, you can grant or deny access to entire applications, such as Pricing Center or Customer Center, or to individual tabs and fields within Customer Center. This enables you to assign CSRs to roles that correspond to their job functions.

For information on how to create roles and permissions and how to assign CSR accounts to roles, see the Permissioning Center Help.

The installation procedure for Permissioning Center is similar to the installation of the other client applications in BRM. See "Installing BRM Client Applications on Windows" in *BRM Installation Guide*.

About Access to Permissioning Center

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.

A person who has BRM **root** privileges has full access permissions to create roles and to add permissions to or delete permissions from a role in Permissioning Center, Customer Center, or Pricing Center.

A brand manager has default access to Permissioning Center. For other client applications, like Customer Center and Pricing Center, brand managers must assign roles that contain permissions for accessing the other client applications to themselves.

Note: After installing Permissioning Center, only the BRM **root** login name has permissions to use Permissioning Center.

For more information on how to manage and include permissions in a role, see the Permissioning Center Help.

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.

For example, CSRs A and B can be assigned to the role Manager, and CSRs C and D can be assigned to the role Lead-CSR, where:

- CSRs A and B have read-write access for customer credit card information.

- CSRs C and D have read-only permissions for customer credit card information.

You can also create roles with higher levels of permissions. For example, you can create roles that include permissions to create and manage roles using Permissioning Center.

About Multiple Role Support

You can include specific permissions in a role to access one or more client applications. In addition, a CSR can be assigned to multiple roles. For example, a CSR can be assigned to a Manager role in Permissioning Center and to a Junior-CSR role in Pricing Center.

About Hierarchical Roles

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. For more information, see ["About Permissioning Center"](#).

In Customer Center, you can provide only one permission at a time to a CSR. Roles cannot be created in Customer Center.

Important: The permissions set in Customer Center overwrite those set in Permissioning Center.

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.

Note: By default, the password status for a brand manager is set to **Normal**, and the status for a CSR account is set to **Temporary**.

When a CSR logs in to a BRM application, the system checks for the password status and takes the following actions:

- If the CSR logs in to the application for the first time or if the password is set to **Temporary**, the Change Password screen is displayed. The CSR is required to change the password before using the application. After the CSR changes the password, the password status is changed from **Temporary** to **Normal**.
- If the password status is set to **Normal**, the CSR password never expires.
- If the password status is set to **Expires**, Customer Center invalidates the CSR password after a specified number of days (90 days by default). When a CSR tries to log in after the expiration time period, the CSR is required to change the password. After the CSR changes the password, the password status remains as **Expires**, and the new password expires after the specified number of days.

For information about setting the default password expiration, see ["Setting the Default Password Expiry Duration"](#).

- If the password status is set to **Invalid**, access to the application is denied.

Automatic Logout

After you log in to a BRM client application and leave it idle for a specified interval, the session expires automatically. To reconnect to the session, you are prompted to enter your password. However, if you have specified the password in the **Infranet.properties** file, the application automatically reads the password from the **Infranet.properties** file and automatically reconnects.

Note: The default interval for automatic logout is set to **0** minutes, which implies that this feature is disabled by default. However, this might be different in your BRM system depending on the number specified in the **cm_timeout** entry in the CM's **pin.conf** file. For more information on **cm_timeout**, see ["Setting the CM Time Interval between Opcode Requests"](#).

To access the application after an automatic logout:

1. In the Confirm Password dialog box, enter the password.

2. Click **OK**.

Note: BRM client applications use multiple connections. Because the interval for automatic logout is different for each connection, you must perform the above steps once for each connection. Also, if you do not access a window or dialog box for a specified interval, the session expires automatically.

Changing the Password

After you log in to a BRM client application, you can change your existing password. After you change the password, you can log in to any BRM application with your new password.

Note:

- You cannot access the application with a temporary or expired password. You are required to change the password to proceed.
- The number of attempts at entering a password is limited. If you exceed this limit, your account is locked and only a system administrator can unlock the account.

The default number of attempts is five, but it might be different in your BRM system depending on the value of the **<MaxLoginAttempts>** element in the **bus_params_act.xml** file. See ["Configuring the Maximum Number of Invalid Login Attempts"](#).

To change the password:

1. From the **File** menu, select **Change Password**.
The Change Password dialog box appears.
2. In the **Enter current password** field, enter your current password.
3. In the **Enter new password** field, enter your new password.
4. In the **Re-enter new password** field, enter the same password to confirm.
5. Click **OK**.

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.

Important:

- The **pin_unlock_service** utility needs a configuration (**pin.conf**) file in the same directory from which you run the utility. See ["Creating Configuration Files for BRM Utilities"](#).
 - 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 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. See ["Starting and Stopping the BRM System"](#).

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/pin@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 **admin_client** or **pcm_client** object. BRM stores the permission settings for each role at the **root** or brand 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.

How Permission Settings Are Stored

You use Permissioning Center to define permission settings for each role at the **root** or brand level in the **/service/admin_client** object in the BRM database.

When you assign a CSR account to a role, the account includes all the permissions that are assigned to the role.

Important: A CSR account can be assigned to only one role per application.

The PIN_FLD_ROLE field in the account contains the roles assigned to the CSR account.

Permission Types

You can set permissions for the following applications:

- Collections Center. See "[Collections Center Permission Types](#)".
- Customer Center. See the Customer Center Help.
- Pricing Center. See "[Pricing Center Permission Types](#)".
- Suspense Management Center. See "[Suspense Management Center Permission Types](#)".
- Other client applications. See "[Other Client Application Permission Types](#)".

Note:

- 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 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.
 - Permission types are case sensitive.
 - The Min value does not apply to any of the Suspense Management Center permission types.
-
-

Customer Center Permission Type

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.

Collections Center Permission Types

Table 5–1 shows the permissions you can set for Collections Center.

Table 5–1 Collections Center Permission Types

Permission Type	Provides Permission To . . .
<code>/appcenter/collectioncenter</code>	Use Collections Center.
<code>/accounttool/collections/agent</code>	Perform the role of <i>collections agent</i> , who carries out collections tasks without any supervisory responsibilities.
<code>/accounttool/collections/manager</code>	Perform the role of <i>collections manager</i> , who supervises collections agents.
<code>/collectionapps/collections/actionhistory</code>	View the history of individual tasks.
<code>/collectionapps/collections/assign</code>	Assign bill units to collections agents.
<code>/collectionapps/collections/changestatus</code>	Change the status of a task.
<code>/collectionapps/collections/chargeCC</code>	Receive credit card payments.
<code>/collectionapps/collections/exempt</code>	Exempt bill units from collections.
<code>/collectionapps/collections/insert</code>	Insert an additional task into a bill unit scenario.
<code>/collectionapps/collections/maskcarddetails</code>	View all of the credit card numbers or all of the direct debit account numbers. Note: If credit card tokenization is enabled, you can view only the last 4 digits of the credit card number. For more information about credit card tokenization, see "About Credit Card Tokenization" in <i>BRM Configuring and Collecting Payments</i> .
<code>/collectionapps/collections/promisetopay</code>	Set the promise-to-pay feature for any bill unit.
<code>/collectionapps/collections/removeexempt</code>	Remove bill units from exemption.
<code>/collectionapps/collections/reschedule</code>	Reschedule tasks.
<code>/collectionapps/collections/updatepayment</code>	Update the payment details for a bill unit.
<code>/collectioncenter/newcard</code>	Register new cards or new accounts.

Pricing Center Permission Types

Table 5–2 shows the permissions you can set for Pricing Center.

Table 5–2 Pricing Center Permission Types

Permission Type	Provides Permission To . . .
<code>/appcenter/pricingcenter</code>	Use Pricing Center.
<code>/appcenter/provisioningtags</code>	Use Provisioning Tags.
<code>/appcenter/ResourceEditor</code>	Use Resource Editor.
<code>/appcenter/ZoneMapper</code>	Use Zone Mapper.

Table 5–2 (Cont.) Pricing Center Permission Types

Permission Type	Provides Permission To . . .
/loadpricelist/access	Load price list XML files into the BRM database.
/pricingcenter/databaseaccess	<p>Take actions that read from and write to the BRM database.</p> <p>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.</p> <p>If this permission type is set to None, you can only work offline in Pricing Center.</p>
/pricingcenter/filesystemaccess	<p>Take actions that read from and write to files.</p> <p>This permission type controls access to the following File menu commands:</p> <p>Open</p> <p>Import - Real-time Data</p> <p>Export - Real-time Data</p> <p>Save</p> <p>Save As</p>

Suspense Management Center Permission Types

Table 5–3 shows the permissions you can set for Suspense Management Center.

Table 5–3 Suspense Management Center Permission Types

Permission Type	Provides Permission To . . .	Max Value Applies
/appcenter/suspensemgt	Use Suspense Management Center.	No
/appcenter/suspensemgt/archive_and_purge	Archive and purge call records, save the records to an archive file, and remove them from the BRM database.	Yes
/appcenter/suspensemgt/batch	Search batches.	Yes
/appcenter/suspensemgt/batch_writeoff	Write off batches.	Yes
/appcenter/suspensemgt/batch_purge	Remove batches from the database.	Yes
/appcenter/suspensemgt/batch_resubmit	Resubmit batches and send them back through a pipeline for rating.	Yes
/appcenter/suspensemgt/bulkedit	Edit a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkpurge	Delete a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkrecycle	Recycle a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkwriteoff	Write off a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/edit	Edit suspended call records.	Yes
/appcenter/suspensemgt/purge	Purge call records from the BRM database.	Yes
/appcenter/suspensemgt/records	Search records.	Yes

Table 5–3 (Cont.) Suspense Management Center Permission Types

Permission Type	Provides Permission To . . .	Max Value Applies
/appcenter/suspensemgt/recycle	Recycle suspended call records and send them back through a pipeline for rating.	Yes
/appcenter/suspensemgt/restore	Restore call records from an archive file.	No
/appcenter/suspensemgt/undo_edit	Undo edits to suspended call records.	No
/appcenter/suspensemgt/writeoff	Write off suspended call records.	Yes

Other Client Application Permission Types

Table 5–4 shows the permissions for accessing client applications. To set up permission for accessing a client application, choose that application's name from the **Application** list and enter the permission type.

Table 5–4 Other Client Application Permission Types

Permission Type	Provides Permission To . . .
/appcenter/collectionconfig	Use Collections Config.
/appcenter/IPAdmin	Use IP Address Administration Center.
/appcenter/paymentcenter	Use Payment Center.
/appcenter/racenter	Use Revenue Assurance.
/paymenttool/access	Use Payment Tool.

Access Levels

The permission levels included in a role determine the access level of the CSR. The permission levels are listed in Table 5–5:

Table 5–5 CSR Permission Levels

Permission Level	Display Data?	Change Data?
None	No	No
Read Only	Yes	No
Write Only Note: This level is typically used only for credit card numbers or passwords. For most permission types, this level is the same as None .	No	Yes
Read/Write	Yes	Yes

Setting Up Access to Brands

BRM systems that use branded service management to host multiple brands of service use privileges to keep each brand's customer accounts and price lists separate. The **root** account has access to all brands. The Brand Manager accounts that you define when creating a brand have access to all data in that brand.

Other users, such as CSRs, get access to brands through access groups. Each access group gives member users access to a single brand or to a single subbrand or account group within a brand.

You create access groups in the Access Privileges application in Configuration Center.

For more information, see "About Granting Access to Brands" in *BRM Managing Customers* and the Access Privileges Help.

Protecting BRM Passwords

BRM stores account passwords (for `/service/admin_client` and `/service/pcm_client`) in encrypted form in the database.

Important: If a BRM user forgets a password, you must provide a new one; you cannot retrieve the old one.

You can change the encryption algorithm, which is AES by default, to another algorithm. To do this, modify the `fm_cust_pol_encrypt_passwd.c` policy module in `BRM_home/source/sys/fm_cust_pol`. See "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide* for more information.

BRM developers can also encrypt other string fields. If your BRM system uses a custom encryption method, the encryption values in the DM configuration file (`BRM_home/sys/dm_oracle/pin.conf`) should be updated. For instructions, see "Encrypting Fields" in *BRM Developer's Guide*.

IP passwords (for `/service/ip`) are clear text; this is a requirement of Challenge Handshake Authentication Protocol (CHAP). However, the password is never sent to the NAS (terminal server), either in encrypted or clear-text form because CHAP encrypts a random-number challenge token based on the customer's password.

Enabling Secure Communication between BRM Components

Secure connections between applications can be obtained by using protocols such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS). The term *SSL* is often used to refer to either of these protocols or a combination of the two (SSL/TLS). BRM supports SSL/TLS to provide secure communication between:

- CMs and DMs
- CMs and External Modules (EMs)

Note: Currently, secure communication using SSL/TLS is not supported between CMs and NET_EM.

- PCM clients and the CM
- PCM clients and the CM when using Connection Manager Master Processes (CMMPs)
- BRM server and BRM database

Note: BRM no longer supports Secure Sockets Layer 3.0. For BRM, Oracle recommends that you use TLS protocol for secure communication. In the BRM 7.5 documentation, secure communications is referred to by the generic term *SSL/TLS*.

To enable secure communication between BRM components:

1. Create an Oracle wallet. See ["Working with SSL/TLS Certificates and Oracle Wallets"](#).
2. Enable SSL/TLS in the CM. See ["Enabling SSL/TLS in Connection Managers"](#).
3. Enable SSL/TLS in the DM. See ["Enabling SSL/TLS in Data Managers"](#).
4. Enable SSL/TLS in your C PCM clients. See ["Enabling SSL/TLS for C and C++ PCM Clients"](#).
5. Enable SSL/TLS in your Java PCM clients. See ["Enabling SSL/TLS for Java PCM Clients"](#).
6. Enable SSL/TLS in your Java server process. See ["Enabling SSL/TLS for Java Server Processes"](#).
7. Enable SSL/TLS in CMMP. See ["Enabling SSL/TLS in Connection Manager Master Processes"](#).
8. Enable SSL/TLS in BRM JCA Resource Adapter. See the description of the `JavaPcmSSL`, `SSLWalletLocation`, and `SslCipherSuites` entries in *BRM JCA Resource Adapter*.

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 provides sample server and client Oracle wallets and trusted certificates that are compatible with the default cipher suite, `SSL_RSA_WITH_AES_128_CBC_SHA`. The sample server and client Oracle wallets and trusted certificates can be used for testing purposes. Set your CM and PCM clients to the following directories:

- To use the server Oracle wallets and trusted certificate samples, set the CM to *BRM_home/wallet/server*. The server Oracle wallet contains the trusted server certificate and a certificate authority (CA) certificate.
- To use the client Oracle wallet and trusted certificate samples, 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"](#).

Note: The password to make changes to the server and client Oracle wallet samples is **Welcome1**.

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 can be found in *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:

```
create_svr_wallet_cert -dn "CN=host_name"
```

where *host_name* is the server host name.

An Oracle wallet and trusted server certificate are created in the *BRM_home/wallet/server* directory. The **create_svr_wallet_cert** script uses the sample CA certificate to sign the server certificate. When asked for the password to make changes to the server Oracle wallet, use **Welcome1**.

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 Welcome1
```

- 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 Welcome1
```

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 Welcome1
```

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 Welcome1
```

- 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*.

Creating an Oracle Wallet

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.
- *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

Caution: 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, which 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.

By default, the following BRM-supported cipher suites are set to use SSL/TLS for C and C++ PCM clients:

- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA256
- SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_AES_128_GCM_SHA256
- SSL_RSA_WITH_AES_256_GCM_SHA384
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

By default, the following BRM-supported cipher suites are set to use SSL/TLS for Java PCM clients:

- TLS_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Enabling SSL/TLS in Connection Managers

By default, SSL/TLS is disabled 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.

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_RC4_128_MD5
```

- 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_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.
7. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).

Enabling SSL/TLS in Data Managers

By default, SSL/TLS is disabled 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.

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:

```
- dm cipher SSL_RSA_WITH_RC4_128_MD5
```

- To add multiple cipher suites, separate each cipher suite by a comma.

For example:

```
- dm cipher 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.
7. Stop and restart the DM. See ["Starting and Stopping the BRM System"](#).

Enabling SSL/TLS for C and C++ PCM Clients

By default, the ability to use SSL/TLS with C and C++ PCM clients is disabled.

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:
3. (Optional) To enable two-way server and client authentication between the CM and C and C++ PCM clients, add the following entry:

```
- nap enable_ssl 1
```

```
- 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.

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_RC4_128_MD5
```

- To add multiple cipher suites, separate each cipher suite by a comma.

For example:

```
- nap cipher 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.
7. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).

Enabling SSL/TLS for Java PCM Clients

By default, the ability to use SSL/TLS with Java PCM clients is disabled. 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.

Important: 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 5–6](#) lists the default location of the **Infranet.properties** file for each Java PCM client.

Table 5–6 BRM Client Infranet.properties File Default Locations

Client Name	Directory Path
Business Configuration Center	Windows XP — C:\Program Files\Common Files\Portal Software\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Revenue Assurance Center	Windows XP — C:\Program Files\Portal Software\RevenueAssuranceCtr\RevenueAssuranceCenter\lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\RevenueAssuranceCtr\RevenueAssuranceCenter\lib\Infranet.properties
Collections Center	Windows XP — C:\Program Files\Portal Software\Collections Center\Lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\Collections Center\Lib\Infranet.properties
Collections Configuration	Windows XP — C:\Program Files\Portal Software\CollectionsConfiguration\Lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\CollectionsConfiguration\Lib\Infranet.properties
Customer Center	Windows XP — C:\Program Files\Portal Software\CustomerCenter\lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\CustomerCenter\lib\Infranet.properties

Table 5–6 (Cont.) BRM Client Infranet.properties File Default Locations

Client Name	Directory Path
Developer Center	Windows XP — C:\Program Files\Common Files\Portal Software\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
IP Address Administration Center	Windows XP — C:\Program Files\Portal Software\IPAddressAdministrationCenter\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\IPAddressAdministrationCenter\Infranet.properties
Number Administration Center	Windows XP — C:\Program Files\Common Files\Portal Software\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Permissioning Center	Windows XP — C:\Program Files\Portal Software\PermissioningCenter\lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\PermissioningCenter\lib\Infranet.properties
Pricing Center	Windows XP — C:\Program Files\Portal Software\PricingCenter\lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\PricingCenter\lib\Infranet.properties
SIM Administration Center	Windows XP — C:\Program Files\Common Files\Portal Software\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Common Files\Portal Software\Infranet.properties
Suspense Management Center	Windows XP — C:\Program Files\Portal Software\SuspenseManagementCenter\lib\Infranet.properties Windows 7 and Windows 8.1 — C:\Program Files (x86)\Portal Software\SuspenseManagementCenter\lib\Infranet.properties
Voucher Administration Center	Windows XP — C:\Program Files\Common Files\Portal Software\Infranet.properties Windows 7 and Windows 8.1 — 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.

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 is 30000.

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=SSL_RSA_WITH_RC4_128_MD5
```

- 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,SSL_RSA_WITH_RC4_128_MD5
```

By default, BRM uses `TLS_RSA_WITH_AES_128_CBC_SHA`. For information on cipher suites supported by BRM, see ["BRM-Supported Cipher Suites"](#).

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*, *BRM_home/jars/httpclient-4.4.jar*, and *BRM_home/jars/commons-logging-1.2.jar* files are in the CLASSPATH.

Enabling SSL for Customer Center Web Start Deployment

To enable SSL for Customer Center Web Start deployment:

1. Package the SSL wallet files into a JAR file; for example, *wallet.jar*.
2. Copy the JAR file to the *Customer_Center_Home/3plibs* directory, where *Customer_Center_Home* is the directory in which you installed Customer Center on your Web server.
3. Create a Java Network Launch Protocol (JNLP) file; for example, *brmwallet.jnlp*, and copy it to the *Customer_Center_Home* directory in which the **`CustomerCenter_en.jnlp`** is located.

The following is the sample 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. Add the JNLP file name in the **resources** section of the *Customer_Center_Home/CustomerCenter_en.jnlp* file. For example:

```

<resources>
    <extension name="brmwallet" href="brmwallet.jnlp"/>
</resources>

```

5. Specify the location and name of the wallet file by adding the following entries in your **Infranet.properties** file packaged in a JAR file, which is specified in the **<resources>** section in the *CustomerCenter_en.jnlp* file:

- **infranet.pcp.ssl.wallet.location=wallet_location/wallet**
- **infranet.pcp.ssl.wallet.filename=wallet_name.sso**

- *wallet_location* is the wallet path within the JAR file that you created in step 1.
- *wallet_name* is the name of the wallet file.

For example:

```

infranet.pcp.ssl.wallet.location=com/oracle/wallet
infranet.pcp.ssl.wallet.filename=cwallet.sso

```

Enabling SSL/TLS for Java Server Processes

By default, the ability to use SSL/TLS with Java server processes (such as the EAI Java Server or *eai_js* and BRM invoice formatter) is disabled in BRM.

Important: 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 5–7](#) lists the default location of the **Infranet.properties** file for each Java server process.

Table 5–7 Java Server Process Infranet.properties File Default Locations

Process Name	Directory Path
EAI Java Server or <i>eai_js</i>	<i>BRM_home/sys/eai_js/Infranet.properties</i>
BRM invoice formatter	<i>BRM_home/sys/formatter/Infranet.properties</i>

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 the Java server process, add the following entry:
 - `infranet.pcp.ssl.client_auth = true`
- 5. (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.
- 6. (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.
- 7. (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=SSL_RSA_WITH_RC4_128_MD5`
 - To add multiple cipher suites, separate each cipher suite by a comma.

For example:

 - `infranet.pcp.ssl.handshake.ciphersuites=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"](#).
- 8. Save and close the file.

Enabling SSL/TLS in Connection Manager Master Processes

By default, SSL/TLS is disabled in CMMPs.

Note: 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"](#).

Note: 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:
3. (Optional) To enable two-way server and client authentication between the CMMP and the PCM client, add the following entry:

- **cm enable_ssl** 1

- **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.

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_RC4_128_MD5

- 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_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.
7. Stop and restart the CMMP. See ["Starting and Stopping the BRM System"](#).

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.

Important: To enable SSL/TLS for Payment Tool in Windows 7 and Windows 8.1:

- 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*).

Important: 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 5–8](#) lists the fields and values that you must set in the login flist to enable SSL/TLS.

Table 5–8 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 .

Table 5–8 (Cont.) Login list Fields for SSL/TLS

Field	Data Type	Value
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.

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

The 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, the administrator verifies 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.

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 5–9](#). 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 5–9](#). 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 5–9 *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.

BRM OMF Instrumented Objects

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Operations Management Framework (OMF) instrumented objects. It also includes a simplified, more readable, version of the BRM Management Information Base (MIB).

For more information, see ["Using the SNMP Instrumentation Protocol to Monitor and Control BRM Components"](#).

About the BRM MIB

The BRM MIB defines the structure of the managed data in the BRM system. All instrumented objects are included in the MIB except for probes that use the Big Table format.

The BRM MIB is described in the **PORTAL-MIB.txt** file in *BRM_home/instrumentation*.

About the Top-Level Components in the MIB

- **log**: Provides recent messages from log files.

Note: These probes are BigTable probes and are therefore not included in the BRM MIB.

- **omf**: Provides data about the SNMP and HTTP OMF protocols and the Diagnostic Data Handler.
- **mm1**: Provides data about the Memory Manager.
- **plg**: Provides data about EDR statistics.
- **fct**: Provides data about opcode statistics.
- **ppl**: Provides data about the Pipeline Manager input controller.

Transaction Manager Probes

The Transaction Manager probes are listed in [Table 6-1](#):

Table 6–1 Transaction Manager Probes

Probe	Description
batchSizeLimit	Minimum size of a batch before committing it to the log file, as configured in the Transaction Manager BatchSizeLimit registry entry.
loggingOff	Indicates if logging is on or off, as configured in the Transaction Manager LoggingOff registry entry.

Log File Probes

For information about the log file OIDs, see ["Log File OIDs"](#).

The probe listed in [Table 6–2](#) collects status and error messages in a circular buffer. It collects all messages, including debugging information, independent of log file settings.

By default, the circular buffer size is set to 1000 messages. You can change that setting in the registry. See ["Getting Recent Pipeline Log File Entries"](#).

You can display this data in the HTTP protocol. It is also included in the file created by the Diagnostic Data Handler.

Note: Log file probes are Big Table probes and are therefore not included in the BRM MIB.

Table 6–2 Log File Probe

Probes	Description
message	Recent log file messages.

Operations Management Framework Probes

Operations Management Framework (OMF) includes the following probes:

- **http**: See ["HTTP Probes"](#).
- **snmp**: See ["SNMP Probes"](#).
- **diagnosticDataHandler**: See ["Diagnostic Data Handler Probe"](#).

For information about the OMF OIDs, see ["OMF OIDs"](#).

HTTP Probes

The HTTP probes are listed in [Table 6–3](#):

Table 6–3 HTTP Probes

Probe	Description
port	The port number for the HTTP listener as configured in the Port registry entry.
styleSheetName	The XML style sheet as configured in the StyleSheetFile registry entry.
portalLogoName	The file name of the logo used on the HTTP display, as defined in the PortalLogoFile registry entry.
numHttpRequests	The number of requests to get data from the HTTP server.

SNMP Probes

The probes listed in [Table 6–4](#) show data about:

- The SNMP instrumentation configuration.
- SNMP activity; for example, the number of GET requests.

The SNMP table includes SNMP activity about specific instrumented objects.

[Figure 6–1](#) shows an example:

Figure 6–1 Sample SNMP Table

Snmp Mib Table Info:

Snmp Mib Table Index	Mib Table Name	Snmp Mib Table Oid	Number of 'GET' Requests	Number of 'GETNEXT' Requests	Number of 'SET' Requests
1	ProcessTable	1.3.6.1.4.1.3512.1.101.1	5	0	0
2	RegistryTable	1.3.6.1.4.1.3512.1.102.1	5	0	0
3	diagnosticDataHandlerGroup	1.3.6.1.4.1.3512.1.4.3.1.1	0	0	1
4	httpInfo	1.3.6.1.4.1.3512.1.4.1.1.1	0	0	0
5	snmpInfo	1.3.6.1.4.1.3512.1.4.2.1.1	0	0	0
6	snmpTable	1.3.6.1.4.1.3512.1.4.2.2.1	0	0	0
7	inputControllerGroup	1.3.6.1.4.1.3512.1.11.1.1.1	6	0	0
8	processingTimeTable	1.3.6.1.4.1.3512.1.11.1.2.1	0	0	0
9	statisticGroup	1.3.6.1.4.1.3512.1.9.1.1.1	0	0	0
10	statisticTable	1.3.6.1.4.1.3512.1.9.1.2.1	4	0	0

Table 6–4 SNMP Probes

Probe	Description
port	The port number for the SNMP subagent as configured in the Port registry entry.
numSnmpRequests	The number of SNMP requests received.
snmpTableEntry	A table showing OMF instrumented objects and SNMP Get, GetNext, and Set requests used for those objects.
snmpTableOid	The OID of the object for which SNMP data is shown.
snmpTableName	The object name in the MIB; for example, ProcessTable , statisticGroup , or diagnosticDataHandlerGroup .
numSnmpGetRequests	The number of SNMP Get requests received.
numSnmpGetNextRequests	The number of SNMP GetNext requests received.
numSnmpSetRequests	The number of SNMP Set requests received.

Diagnostic Data Handler Probe

Use the probe listed in [Table 6–5](#) to get a snapshot from the Diagnostic Data Handler.

For information about the Diagnostic Data Handler, see ["Using the Diagnostic Data Handler to Get OMF Diagnostic Data"](#).

Table 6–5 Diagnostic Data Handler Probe

Probe	Description
createSnapshot	Creates a Diagnostic Data Handler snapshot. To get a snapshot, set to true : Use all lowercase characters. Note: This probe is a write-only probe. It is not displayed in the HTTP interface.

Multi-Threaded Framework

Multi-threaded framework (MTF) includes the following probes:

- **connectionConfigurations**: See ["Connection Configurations Probes"](#).
- **threadCheckManager**: See ["Thread Check Manager Probes"](#).
- **stateManager**: See ["MTF State Manager Probes"](#).

For information about the MTF OIDs, see ["MTF OIDs"](#).

Connection Configurations Probes

The connection configurations probes listed in [Table 6–6](#) include:

- **connectionConfigurationsGroup**: These probes provide data about the Oracle DM configuration settings. See ["Connection Configurations Probes"](#).
- **dmoTable**: These probes provide data about the configured Oracle DMs. See ["Oracle DM Server Probes"](#).

Connection Configurations Probes

The probes shown in [Figure 6–2](#) and listed in [Table 6–6](#) provide data about the available Oracle DM servers.

Figure 6–2 Connection Configuration Probes

Connection Configurations:

Total number of configured DMO Servers	1
Number of available DMO Servers	1
Peer Configured	yes

Table 6–6 Connection Configurations Probes

Probe	Description
dmoServerConfigured	The number of Oracle DM servers configured.

Table 6–6 (Cont.) Connection Configurations Probes

Probe	Description
dmoServerAvailable	The number of available Oracle DM servers.
peerConfigured	Specifies if a peer is configured.
addServerInstance	Rebalances DM connections when you restart a failed DM or when you add a new DM to the DM pool. When you use this to add a new DM, provide the host name, port number, and maximum connections it can process. Note: This probe is a write-only probe. It is not displayed in the HTTP interface.

Oracle DM Server Probes

The probes shown in [Figure 6–3](#) and listed in [Table 6–7](#) provide data about each Oracle DM.

Figure 6–3 Oracle DM Server Probes**DMO Server Table:**

DMO Server	DMO Server Host Name	DMO Server Port Number	Maximum Connections	Active Connections
1	sunra	13008	64	11

Table 6–7 Oracle DM Server Probes

Probe	Description
dmoHost	The Oracle DM host name.
dmoPort	The Oracle DM port number.
dmoMaxConn	The number of maximum connections that the Oracle DM can handle at any time.
dmoActiveConn	Active connections for this Oracle DM.

Thread Check Manager Probes

Thread Check Manager monitors threads and reports failures as part of a high-availability configuration.

Thread Check Manager Probes

The probe listed in [Table 6–8](#) shows the current **ThreadCheckManager** registry settings.

Table 6–8 Thread Check Manager Probes

Probe	Description
requestInterval	The time in seconds to poll for thread responsiveness.

Thread Chart Probes

The probes shown in [Figure 6–4](#) and listed in [Table 6–9](#) provide data about active threads.

Figure 6–4 Thread Chart Probes**Thread Chart Table:**

Thread Chart Table Index	Name	Id	Status	Count	Time last updated	Message of last update
1	timosMgr.TimosDataManager.AcceptorPool	14	HEALTHY	22	06/20/06 12:02:59	ClientConnection
2	timosMgr.TimosDataManager.ConnectorPool	23	HEALTHY	22	06/20/06 12:02:59	Default Gateway
3	timosMgr.TimosDataManager.AcceptorPool	22	HEALTHY	22	06/20/06 12:02:59	ClientGateway
4	timosMgr.TimosDataManager.AcceptorPool	21	HEALTHY	22	06/20/06 12:03:00	ClientConnection

Table 6–9 Thread Chart Probes

Probe	Description
threadChartThreadName	The name of the thread.
threadChartThreadId	The thread ID.
threadChartStatus	The status, HEALTHY or UNHEALTHY .
threadChartTime	The timestamp when the thread data was collected.
threadChartCount	The number of times the thread has been checked for responsiveness.
threadChartMessage	The type of thread being monitored: <ul style="list-style-type: none"> ClientConnection ClientGateway DataMigrator WorkerDataMigratorThread PoidIdManager

MTF State Manager Probes

The probes shown in [Figure 6–5](#) and listed in [Table 6–10](#) provide data about the MTF framework.

Figure 6–5 MTF State Manager Probes**MTF State Manager:**

State	Running
Number of open connections	0

Table 6–10 MTF State Manager Probes

Probe	Description
state	<p>The current execution state of the MTF framework:</p> <ul style="list-style-type: none"> ■ Initial: The system is starting up. ■ Running: Normal operation. The MTF framework is accepting client requests. ■ SuspendInProgress: The MTF framework is in the process of shutting down client connections. No new connections are allowed in this state. ■ Suspended: The MTF framework is not accepting connections so that a shutdown or switchover can occur.
numOfConnection	The number of current client connections.

AAA Manager Connection and Number Portability Probes

Use these probes to manage AAA Manager connections and number portability.

- **connectionMonitor:** See ["Connection Monitor Probes"](#).
- **numberPortability:** These probes provide data for number portability. See ["Number Portability Probes"](#).

Connection Monitor Probes

Use the probe listed in [Table 6–11](#) to send a Diameter Disconnect Peer Request (DPR) message.

For information about the connection monitor OIDs, see ["AAA Manager Connection and Number Portability OIDs"](#).

Table 6–11 Connection Monitor Probe

Probe	Description
sendDPR	Sends a Disconnect Peer Request (DPR) message.

Number Portability Probes

Use the probes listed in [Table 6–12](#) to print, load, and append the number portability records. See "Setting Up Number Portability" in *BRM Configuring Pipeline Rating and Discounting*.

For more information about the number portability OIDs, see ["AAA Manager Connection and Number Portability OIDs"](#).

Table 6–12 Number Portability Probes

Probe	Description
reload	Reloads the data from the number portability file. Value: True or False . Set to True to continue to use the old number portability data if the reload operation fails.
deltaLoad	Appends the additional number portability data in the memory. Location: <i>Pipeline_home</i> Value: <i>File_name</i> Where <i>File_name</i> is the name of the delta file.
printData	Displays the in-memory number portability data or saves it in a file. Location: <i>Pipeline_home</i> Value: NULL or <i>File_name</i> Set to NULL to display the number portability data on the screen. Set to the <i>File_name</i> to save the number portability data in the file.

Memory Monitor Probes

The probe listed in [Table 6–13](#) provides data on memory allocation.

For information about the memory monitor OIDs, see "[Memory Monitor OIDs](#)".

Table 6–13 Memory Monitor Probes

Probe	Description
totalMemoryAllocated	The total memory allocated to the process.

Pipeline Statistics Probes

You can get EDR throughput statistics for individual pipelines.

For information about the pipeline statistics OIDs, see "[Statistics OIDs](#)".

Throughput Statistics Probes

The probes shown in [Figure 6–6](#) and listed in [Table 6–14](#) provide data about the total EDRs processed by a pipeline.

Figure 6–6 Throughput Statistics Probes

Throughput Statistics:

Total EDR Count (real-time)	210
Total EDR Count (after transaction ended)	120
Accumulated Txn Processing Time (sec)	3
Total Txn Count	40

Table 6–14 Throughput Statistics Probes

Probe	Description
totalEdrCount	The total number of EDRs processed since startup, independent of any transaction, including EDRs processed in canceled and rolled-back transactions. This value is incremented for each EDR.
totalTxnEdrCount	The total number of EDRs processed in a transaction. This value is incremented after a transaction finishes.
totalProcTime	The total processing time in seconds since startup.
totalTxnCount	The number of transactions processed since startup.

Last Throughputs Probes

The probes shown in [Figure 6–7](#) and listed in [Table 6–15](#) provide data about EDR throughput (in a transaction) for a pipeline.

Figure 6–7 Last Throughputs Probes

Last Throughputs:

Index	Throughput (edrs/sec)	Timestamp
1	1053	16.06.2006 11:49:21
2	1025	16.06.2006 11:49:21
3	1076	16.06.2006 11:49:22
4	1055	16.06.2006 11:49:22
5	1476	16.06.2006 11:49:22
6	1057	16.06.2006 11:49:22
7	948	16.06.2006 11:49:22
8	1149	16.06.2006 11:49:23
9	1439	16.06.2006 11:49:23
10	1147	16.06.2006 11:49:23

Table 6–15 Last Throughputs Probes

Probe	Description
throughput	The number of EDRs per second.
throughputTimestamp	The timestamp when the throughput measurement occurred.

Opcode Latency Probes

These probes provide data about the performance of the FCT_Opcode module.

For information about the opcode latency OIDs, see ["fct Probe OIDs"](#).

Opcode Plug-In Probes

The probes listed in [Table 6–16](#) provide statistics on the **opcodePlugInGroup** probes.

Table 6–16 Opcode Plug-In Probes

Probe	Description
totalOpcodeCalls	The total number of opcode calls since startup.
totalLatency	The total amount of time for all opcode calls.

Last Latency Probes

The probes listed in [Table 6–17](#) provide data on opcode latency.

Table 6–17 Last Latency Probes

Probe	Description
latency	The processing time for opcodes.
latencyTimestamp	The timestamp when the latency measurement occurred.

Input Probes

These probes provide data about pipeline input processing.

For information about the input OIDs, see "[Input Controller probes](#)".

Input Controller Probes

The probes in [Figure 6–8](#) and listed in [Table 6–18](#) show processing time for real-time pipeline events.

Figure 6–8 Input Controller Probes

Input Controller:

Maximum Processing Time (ns)	3428982 (16.06.2006 11:49:18)
Minimum Processing Time (ns)	1091240 (16.06.2006 11:49:18)

Table 6–18 Input Controller Probes

Probe	Description
maxProcessingTime	The longest processing time for a real-time event.
minProcessingTime	The shortest processing time for a real-time event.

Last Processing Times Probes

The probes in [Figure 6–9](#) and listed in [Table 6–19](#) show pipeline processing times for the last 10 real-time events.

Figure 6–9 Last Processing Times Probes**Last Processing Times:**

Index	Processing Time (ns)	Timestamp
1	1689462	16.06.2006 11:49:17
2	2294602	16.06.2006 11:49:17
3	1172088	16.06.2006 11:49:17
4	1503916	16.06.2006 11:49:17
5	1909107	16.06.2006 11:49:17
6	2264288	16.06.2006 11:49:17
7	1091240	16.06.2006 11:49:18
8	1505161	16.06.2006 11:49:18
9	2348588	16.06.2006 11:49:18
10	3428982	16.06.2006 11:49:18

Table 6–19 Last Processing Times Probes

Probe	Description
processingTime	The processing time for a real-time event.
timestamp	The timestamp when the processing time was measured.

SNMP Object IDs

The following sections describe the Portal object IDs (POIDs) in order of the ID number. The same information is available in the BRM MIB, but not in order. However, the BRM MIB includes details about every instrumented object; for example:

```
-- 1.3.6.1.4.1.3512.1.5.5.1.1.2
numOfConnection OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION  "Number of open connections"
    ::= { stateManagerGroupEntry 2 }
```

To derive the complete OIDs, in the following sections, replace the:

- *process_id* with the process ID of the SNMP subagent process that is registered with the SNMP master-agent.
- *registry_id* with the registry ID of the SNMP subagent process that is registered with the SNMP master-agent.

For more information, see ["About Dynamic Object IDs"](#).

MIB Prefix

The BRM MIB uses the industry-standard **internet.private** root, with 3512 as the start of the BRM subroot. The only branch following **portal (3512)** is **components (1)**, so the prefix for all entries is:

iso.org.dod.internet.private.enterprise.portal.components
or:
1.3.6.1.4.1.3512.1

Transaction Manager OIDs

txm
1.3.6.1.4.1.3512.1.2

transactionManager
1.3.6.1.4.1.3512.1.2.1

transactionManagerGroup
1.3.6.1.4.1.3512.1.2.1.1

transactionManagerGroupEntry
1.3.6.1.4.1.3512.1.2.1.1.1

batchSizeLimit
1.3.6.1.4.1.3512.1.2.1.1.1.1

loggingOff
1.3.6.1.4.1.3512.1.2.1.1.1.2

transactionManagerGroupIndex
1.3.6.1.4.1.3512.1.2.1.1.1.3

Log File OIDs

Because log file probes use the Big Table format, they are not included in the MIB file.

log
1.3.6.1.4.1.3512.1.3

OMF OIDs

omf
1.3.6.1.4.1.3512.1.4

http
1.3.6.1.4.1.3512.1.4.1

httpInfo
1.3.6.1.4.1.3512.1.4.1.1

httpInfoEntry
1.3.6.1.4.1.3512.1.4.1.1.1

port
1.3.6.1.4.1.3512.1.4.1.1.1.1

styleSheetName
1.3.6.1.4.1.3512.1.4.1.1.1.2

portalLogoName
1.3.6.1.4.1.3512.1.4.1.1.1.3


```
numHttpRequests
1.3.6.1.4.1.3512.1.4.1.1.1.4

httpInfoIndex
1.3.6.1.4.1.3512.1.4.1.1.1.5

snmp
1.3.6.1.4.1.3512.1.4.2

    snmpInfo
    1.3.6.1.4.1.3512.1.4.2.1

        snmpInfoEntry
        1.3.6.1.4.1.3512.1.4.2.1.1

            port
            1.3.6.1.4.1.3512.1.4.2.1.1.1

            numSnmpRequests
            1.3.6.1.4.1.3512.1.4.2.1.1.2

            snmpInfoIndex
            1.3.6.1.4.1.3512.1.4.2.1.1.3

snmpTable
1.3.6.1.4.1.3512.1.4.2.2

    snmpTableEntry
    1.3.6.1.4.1.3512.1.4.2.2.1

        snmpTableIndex
        1.3.6.1.4.1.3512.1.4.2.2.1.1

        snmpTableOid
        1.3.6.1.4.1.3512.1.4.2.2.1.2

        snmpTableName
        1.3.6.1.4.1.3512.1.4.2.2.1.3

        numSnmpGetRequests
        1.3.6.1.4.1.3512.1.4.2.2.1.4

        numSnmpGetNextRequests
        1.3.6.1.4.1.3512.1.4.2.2.1.5

        numSnmpSetRequests
        1.3.6.1.4.1.3512.1.4.2.2.1.6

diagnosticDataHandler
1.3.6.1.4.1.3512.1.4.3

    diagnosticDataHandlerGroup
    1.3.6.1.4.1.3512.1.4.3.1

        diagnosticDataHandlerGroupEntry
        1.3.6.1.4.1.3512.1.4.3.1.1

            createSnapshot
```

1.3.6.1.4.1.3512.1.4.3.1.1.1

diagnosticDataHandlerGroupIndex

1.3.6.1.4.1.3512.1.4.3.1.1.2

MTF OIDs

mtf

1.3.6.1.4.1.3512.1.5

highAvailabilityManager

1.3.6.1.4.1.3512.1.5.1

connectionConfigurations

1.3.6.1.4.1.3512.1.5.2

connectionConfigurationsGroup

1.3.6.1.4.1.3512.1.5.2.1

connectionConfigurationsGroupEntry

1.3.6.1.4.1.3512.1.5.2.1.1

dmoServerConfigured

1.3.6.1.4.1.3512.1.5.2.1.1.1

dmoServerAvailable

1.3.6.1.4.1.3512.1.5.2.1.1.2

peerConfigured

1.3.6.1.4.1.3512.1.5.2.1.1.3

refreshConnections

1.3.6.1.4.1.3512.1.5.2.1.1.4

addServerInstance

1.3.6.1.4.1.3512.1.5.2.1.1.5

connectionConfigurationsGroupIndex

1.3.6.1.4.1.3512.1.5.2.1.1.6

dmoTable

1.3.6.1.4.1.3512.1.5.2.2

dmoEntry

1.3.6.1.4.1.3512.1.5.2.2.1

dmoIndex

1.3.6.1.4.1.3512.1.5.2.2.1.1

dmoHost

1.3.6.1.4.1.3512.1.5.2.2.1.2

dmoPort

1.3.6.1.4.1.3512.1.5.2.2.1.3

dmoMaxConn

1.3.6.1.4.1.3512.1.5.2.2.1.4

```

                                dmoActiveConn
                                1.3.6.1.4.1.3512.1.5.2.2.1.5

                                dmoState
                                1.3.6.1.4.1.3512.1.5.2.2.1.6

threadCheckManager
1.3.6.1.4.1.3512.1.5.3

    threadCheckManagerGroup
    1.3.6.1.4.1.3512.1.5.3.1

        threadCheckManagerGroupEntry
        1.3.6.1.4.1.3512.1.5.3.1.1

            requestInterval
            1.3.6.1.4.1.3512.1.5.3.1.1.1

            notifyFailure
            1.3.6.1.4.1.3512.1.5.3.1.1.2

            threadCheckManagerGroupIndex
            1.3.6.1.4.1.3512.1.5.3.1.1.3

threadCheckManagerTable
1.3.6.1.4.1.3512.1.5.3.2

    threadChartTableEntry
    1.3.6.1.4.1.3512.1.5.3.2.1

        threadChartTableIndex
        1.3.6.1.4.1.3512.1.5.3.2.1.1

        threadChartThreadName
        1.3.6.1.4.1.3512.1.5.3.2.1.2

        threadChartThreadId
        1.3.6.1.4.1.3512.1.5.3.2.1.3

        threadChartStatus
        1.3.6.1.4.1.3512.1.5.3.2.1.4

        threadChartTime
        1.3.6.1.4.1.3512.1.5.3.2.1.5

        threadChartCount
        1.3.6.1.4.1.3512.1.5.3.2.1.6

        threadChartMessage
        1.3.6.1.4.1.3512.1.5.3.2.1.7

threadPool
1.3.6.1.4.1.3512.1.5.4

    threadPoolAcceptorGroup
    1.3.6.1.4.1.3512.1.5.4.1

        threadPoolAcceptorGroupEntry
        1.3.6.1.4.1.3512.1.5.4.1.1
```

```
        acceptorThreadPoolType
        1.3.6.1.4.1.3512.1.5.4.1.1.1

        acceptorThreadCount
        1.3.6.1.4.1.3512.1.5.4.1.1.2

        acceptorReactorType
        1.3.6.1.4.1.3512.1.5.4.1.1.3

        threadPoolAcceptorGroupIndex
        1.3.6.1.4.1.3512.1.5.4.1.1.4

threadPoolAcceptorTable
1.3.6.1.4.1.3512.1.5.4.2

        threadPoolAcceptorTableEntry
        1.3.6.1.4.1.3512.1.5.4.2.1

        acceptorConnectionThreadNumber
        1.3.6.1.4.1.3512.1.5.4.2.1.1

        acceptorAccumulatedConnectionCount
        1.3.6.1.4.1.3512.1.5.4.2.1.2

        acceptorCurrentConnectionCount
        1.3.6.1.4.1.3512.1.5.4.2.1.3

threadPoolConnectorGroup
1.3.6.1.4.1.3512.1.5.4.3

        threadPoolConnectorGroupEntry
        1.3.6.1.4.1.3512.1.5.4.3.1

        connectorThreadPoolType
        1.3.6.1.4.1.3512.1.5.4.3.1.1

        connectorThreadCount
        1.3.6.1.4.1.3512.1.5.4.3.1.2

        connectorReactorType
        1.3.6.1.4.1.3512.1.5.4.3.1.3

        threadPoolConnectorGroupIndex
        1.3.6.1.4.1.3512.1.5.4.3.1.4

threadPoolConnectorTable
1.3.6.1.4.1.3512.1.5.4.4

        threadPoolConnectorTableEntry
        1.3.6.1.4.1.3512.1.5.4.4.1

        connectorConnectionThreadNumber
        1.3.6.1.4.1.3512.1.5.4.4.1.1

        connectorAccumulatedConnectionCount
        1.3.6.1.4.1.3512.1.5.4.4.1.2

        connectorCurrentConnectionCount
        1.3.6.1.4.1.3512.1.5.4.4.1.3
```

threadPoolPrivatePeerGroup
1.3.6.1.4.1.3512.1.5.4.5

threadPoolPrivatePeerGroupEntry
1.3.6.1.4.1.3512.1.5.4.5.1

privatePeerThreadPoolType
1.3.6.1.4.1.3512.1.5.4.5.1.1

privatePeerThreadCount
1.3.6.1.4.1.3512.1.5.4.5.1.2

privatePeerReactorType
1.3.6.1.4.1.3512.1.5.4.5.1.3

threadPoolPrivatePeerGroupIndex
1.3.6.1.4.1.3512.1.5.4.5.1.4

threadPoolPrivatePeerTable
1.3.6.1.4.1.3512.1.5.4.6

threadPoolPrivatePeerTableEntry
1.3.6.1.4.1.3512.1.5.4.6.1

privatePeerConnectionThreadNumber
1.3.6.1.4.1.3512.1.5.4.6.1.1

privatePeerAccumulatedConnectionCount
1.3.6.1.4.1.3512.1.5.4.6.1.2

privatePeerCurrentConnectionCount
1.3.6.1.4.1.3512.1.5.4.6.1.3

qosOpcodeConfigTable
1.3.6.1.4.1.3512.1.5.4.7

qosOpcodeConfigTableEntry
1.3.6.1.4.1.3512.1.5.4.7.1

qosOpcodeConfigIndex
1.3.6.1.4.1.3512.1.5.4.7.1.1

qosOpcodeConfigName
1.3.6.1.4.1.3512.1.5.4.7.1.2

qosOpcodeConfigBucket1
1.3.6.1.4.1.3512.1.5.4.7.1.3

qosOpcodeConfigBucket2
1.3.6.1.4.1.3512.1.5.4.7.1.4

qosOpcodeConfigBucket3
1.3.6.1.4.1.3512.1.5.4.7.1.5

qosOpcodeConfigBucket4
1.3.6.1.4.1.3512.1.5.4.7.1.6

qosOpcodeConfigBucket5
1.3.6.1.4.1.3512.1.5.4.7.1.7

qosOpcodeConfigBucket6
1.3.6.1.4.1.3512.1.5.4.7.1.8

qosOpcodeConfigBucket7
1.3.6.1.4.1.3512.1.5.4.7.1.9

qosOpcodeConfigBucket8
1.3.6.1.4.1.3512.1.5.4.7.1.10

qosOpcodeTable
1.3.6.1.4.1.3512.1.5.4.8

qosOpcodeTableEntry
1.3.6.1.4.1.3512.1.5.4.8.1

qosOpcodeIndex
1.3.6.1.4.1.3512.1.5.4.8.1.1

qosOpcodeName
1.3.6.1.4.1.3512.1.5.4.8.1.2

qosTotalSamples
1.3.6.1.4.1.3512.1.5.4.8.1.3

qosAverageProcessingTime
1.3.6.1.4.1.3512.1.5.4.8.1.4

qosMinProcessingTime
1.3.6.1.4.1.3512.1.5.4.8.1.5

qosMaxProcessingTime
1.3.6.1.4.1.3512.1.5.4.8.1.6

qosThroughput
1.3.6.1.4.1.3512.1.5.4.8.1.7

qosOpcodeBucketCount1
1.3.6.1.4.1.3512.1.5.4.8.1.8

qosOpcodeBucketPercent1
1.3.6.1.4.1.3512.1.5.4.8.1.9

qosOpcodeBucketCount2
1.3.6.1.4.1.3512.1.5.4.8.1.10

qosOpcodeBucketPercent2
1.3.6.1.4.1.3512.1.5.4.8.1.11

qosOpcodeBucketCount3
1.3.6.1.4.1.3512.1.5.4.8.1.12

qosOpcodeBucketPercent3
1.3.6.1.4.1.3512.1.5.4.8.1.13

qosOpcodeBucketCount4
1.3.6.1.4.1.3512.1.5.4.8.1.14

qosOpcodeBucketPercent4
1.3.6.1.4.1.3512.1.5.4.8.1.15

```
qosOpcodeBucketCount5
1.3.6.1.4.1.3512.1.5.4.8.1.16

qosOpcodeBucketPercent5
1.3.6.1.4.1.3512.1.5.4.8.1.17

qosOpcodeBucketCount6
1.3.6.1.4.1.3512.1.5.4.8.1.18

qosOpcodeBucketPercent6
1.3.6.1.4.1.3512.1.5.4.8.1.19

qosOpcodeBucketCount7
1.3.6.1.4.1.3512.1.5.4.8.1.20

qosOpcodeBucketPercent7
1.3.6.1.4.1.3512.1.5.4.8.1.21

qosOpcodeBucketCount8
1.3.6.1.4.1.3512.1.5.4.8.1.22

qosOpcodeBucketPercent8
1.3.6.1.4.1.3512.1.5.4.8.1.23

qosResetTable
1.3.6.1.4.1.3512.1.5.4.9

    qosResetEntry
    1.3.6.1.4.1.3512.1.5.4.9.1

        qosReset
        1.3.6.1.4.1.3512.1.5.4.9.1.1

        qosResetIndex
        1.3.6.1.4.1.3512.1.5.4.9.1.2

threadPoolStatisticsGroup
1.3.6.1.4.1.3512.1.5.4.10

    threadPoolStatisticsGroupEntry
    1.3.6.1.4.1.3512.1.5.4.10.1

        poolTotalRequestCount
        1.3.6.1.4.1.3512.1.5.4.10.1.1

        poolThroughput
        1.3.6.1.4.1.3512.1.5.4.10.1.2

        threadPoolStatisticsGroupIndex
        1.3.6.1.4.1.3512.1.5.4.10.1.3

perThreadStatisticsTable
1.3.6.1.4.1.3512.1.5.4.11

    perThreadStatisticsTableEntry
    1.3.6.1.4.1.3512.1.5.4.11.1

        threadIndex
        1.3.6.1.4.1.3512.1.5.4.11.1.1
```

threadTotalRequestCount
1.3.6.1.4.1.3512.1.5.4.11.1.2

threadThroughput
1.3.6.1.4.1.3512.1.5.4.11.1.3

threadOverloadIntervalRequestCount
1.3.6.1.4.1.3512.1.5.4.11.1.4

overloadDetectionConfigGroup
1.3.6.1.4.1.3512.1.5.4.12

overloadDetectionConfigGroupEntry
1.3.6.1.4.1.3512.1.5.4.12.1

overloadDetectionEnabled
1.3.6.1.4.1.3512.1.5.4.12.1.1

overloadDetectionRate
1.3.6.1.4.1.3512.1.5.4.12.1.2

overloadDetectionInterval
1.3.6.1.4.1.3512.1.5.4.12.1.3

overloadDetectionThreshold
1.3.6.1.4.1.3512.1.5.4.12.1.4

overloadDetectionConfigGroupIndex
1.3.6.1.4.1.3512.1.5.4.12.1.5

stateManager
1.3.6.1.4.1.3512.1.5.5

stateManagerGroup
1.3.6.1.4.1.3512.1.5.5.1

stateManagerGroupEntry
1.3.6.1.4.1.3512.1.5.5.1.1

state
1.3.6.1.4.1.3512.1.5.5.1.1.1

numOfConnection
1.3.6.1.4.1.3512.1.5.5.1.1.2

stateManagerGroupIndex
1.3.6.1.4.1.3512.1.5.5.1.1.3

Data Object OIDs

oal
1.3.6.1.4.1.3512.1.6

storageManager
1.3.6.1.4.1.3512.1.6.1

objectTable
1.3.6.1.4.1.3512.1.6.1.1


```
objectTableEntry
1.3.6.1.4.1.3512.1.6.1.1.1

    objectIndex
    1.3.6.1.4.1.3512.1.6.1.1.1.1

    objectName
    1.3.6.1.4.1.3512.1.6.1.1.1.2

    objectCount
    1.3.6.1.4.1.3512.1.6.1.1.1.3

    objectAvgSize
    1.3.6.1.4.1.3512.1.6.1.1.1.4

indexController
1.3.6.1.4.1.3512.1.6.2

    indexControllerTable
    1.3.6.1.4.1.3512.1.6.2.1

        indexTableEntry
        1.3.6.1.4.1.3512.1.6.2.1.1

            tableIndex
            1.3.6.1.4.1.3512.1.6.2.1.1.1

            indexType
            1.3.6.1.4.1.3512.1.6.2.1.1.2

            unique
            1.3.6.1.4.1.3512.1.6.2.1.1.3

            indexName
            1.3.6.1.4.1.3512.1.6.2.1.1.4

            reverseKey
            1.3.6.1.4.1.3512.1.6.2.1.1.5

            totalInsert
            1.3.6.1.4.1.3512.1.6.2.1.1.6

            insertCollision
            1.3.6.1.4.1.3512.1.6.2.1.1.7

            collisionRatio
            1.3.6.1.4.1.3512.1.6.2.1.1.8

            bucketInstrumentation
            1.3.6.1.4.1.3512.1.6.2.1.1.9

            zeroBucket
            1.3.6.1.4.1.3512.1.6.2.1.1.10

            oneBucket
            1.3.6.1.4.1.3512.1.6.2.1.1.11

            twoBucket
            1.3.6.1.4.1.3512.1.6.2.1.1.12
```

threeBucket
1.3.6.1.4.1.3512.1.6.2.1.1.13

fourBucket
1.3.6.1.4.1.3512.1.6.2.1.1.14

fiveBucket
1.3.6.1.4.1.3512.1.6.2.1.1.15

sixEightBucket
1.3.6.1.4.1.3512.1.6.2.1.1.16

nineSixteenBucket
1.3.6.1.4.1.3512.1.6.2.1.1.17

overSixteenBucket
1.3.6.1.4.1.3512.1.6.2.1.1.18

longestSize
1.3.6.1.4.1.3512.1.6.2.1.1.19

longestBucket
1.3.6.1.4.1.3512.1.6.2.1.1.20

poidIndexController
1.3.6.1.4.1.3512.1.6.2.3

poidIndexControllerTable
1.3.6.1.4.1.3512.1.6.3.1

poidIndexTableEntry
1.3.6.1.4.1.3512.1.6.3.1.1

tableIndex
1.3.6.1.4.1.3512.1.6.3.1.1.1

poidTypeName
1.3.6.1.4.1.3512.1.6.3.1.1.2

poidNumber
1.3.6.1.4.1.3512.1.6.3.1.1.3

residency
1.3.6.1.4.1.3512.1.6.3.1.1.4

lockWaits
1.3.6.1.4.1.3512.1.6.3.1.1.5

totalInsert
1.3.6.1.4.1.3512.1.6.3.1.1.6

insertCollision
1.3.6.1.4.1.3512.1.6.3.1.1.7

collisionRatio
1.3.6.1.4.1.3512.1.6.3.1.1.8

zeroBucket
1.3.6.1.4.1.3512.1.6.3.1.1.9

```
oneBucket
1.3.6.1.4.1.3512.1.6.3.1.1.10

twoBucket
1.3.6.1.4.1.3512.1.6.3.1.1.11

threeBucket
1.3.6.1.4.1.3512.1.6.3.1.1.12

fourBucket
1.3.6.1.4.1.3512.1.6.3.1.1.13

fiveBucket
1.3.6.1.4.1.3512.1.6.3.1.1.14

sixEightBucket
1.3.6.1.4.1.3512.1.6.3.1.1.15

nineSixteenBucket
1.3.6.1.4.1.3512.1.6.3.1.1.16

overSixteenBucket
1.3.6.1.4.1.3512.1.6.3.1.1.17

longestSize
1.3.6.1.4.1.3512.1.6.3.1.1.18

longestBucket
1.3.6.1.4.1.3512.1.6.3.1.1.19
```

AAA Manager Connection and Number Portability OIDs

```
dat
1.3.6.1.4.1.3512.1.7

    connectionPool
    1.3.6.1.4.1.3512.1.7.1

        connectionPoolGroup
        1.3.6.1.4.1.3512.1.7.1.1

            connectionPoolGroupEntry
            1.3.6.1.4.1.3512.1.7.1.1.1

                rebalanceConnection
                1.3.6.1.4.1.3512.1.7.1.1.1.1

                    connectionPoolGroupIndex
                    1.3.6.1.4.1.3512.1.7.1.1.1.2

connectionMonitor
1.3.6.1.4.1.3512.1.7.2

    connectionMonitorGroup
    1.3.6.1.4.1.3512.1.7.2.1

        connectionMonitorGroupEntry
        1.3.6.1.4.1.3512.1.7.2.1.1
```

sendDPR
1.3.6.1.4.1.3512.1.7.2.1.1.1

connectionMonitorGroupIndex
1.3.6.1.4.1.3512.1.7.2.1.1.2

numberPortability
1.3.6.1.4.1.3512.1.7.3

numberPortabilityGroup
1.3.6.1.4.1.3512.1.7.3.1

NumberPortabilityGroupEntry
1.3.6.1.4.1.3512.1.7.3.1.1

reload
1.3.6.1.4.1.3512.1.7.3.1.1.1

deltaLoad
1.3.6.1.4.1.3512.1.7.3.1.1.2

printData
1.3.6.1.4.1.3512.1.7.3.1.1.3

Memory Monitor OIDs

mm1
1.3.6.1.4.1.3512.1.8

memoryManager
1.3.6.1.4.1.3512.1.8.1

memoryManagerGroup
1.3.6.1.4.1.3512.1.8.1.1

memoryManagerGroupEntry
1.3.6.1.4.1.3512.1.8.1.1.1

totalMemoryAllocated
1.3.6.1.4.1.3512.1.8.1.1.1.1

memoryManagerGroupIndex
1.3.6.1.4.1.3512.1.8.1.1.1.2

Statistics OIDs

plg
1.3.6.1.4.1.3512.1.9

statistic
1.3.6.1.4.1.3512.1.9.1

statisticGroup
1.3.6.1.4.1.3512.1.9.1.1

statisticGroupEntry
1.3.6.1.4.1.3512.1.9.1.1.1

```

totalEdrCount
1.3.6.1.4.1.3512.1.9.1.1.1.1

totalTxnEdrCount
1.3.6.1.4.1.3512.1.9.1.1.1.2

totalProcTime
1.3.6.1.4.1.3512.1.9.1.1.1.3

totalTxnCount
1.3.6.1.4.1.3512.1.9.1.1.1.4

statisticGroupIndex
1.3.6.1.4.1.3512.1.9.1.1.1.5

statisticTable
1.3.6.1.4.1.3512.1.9.1.2

statisticEntry
1.3.6.1.4.1.3512.1.9.1.2.1

statisticTableIndex
1.3.6.1.4.1.3512.1.9.1.2.1.1

throughput
1.3.6.1.4.1.3512.1.9.1.2.1.2

throughputTimestamp
1.3.6.1.4.1.3512.1.9.1.2.1.3

```

fct Probe OIDs

```

fct
1.3.6.1.4.1.3512.1.10

opcodePlugIn
1.3.6.1.4.1.3512.1.10.1

opcodePlugInGroup
1.3.6.1.4.1.3512.1.10.1.1

opcodePlugInGroupEntry
1.3.6.1.4.1.3512.1.10.1.1.1

totalOpcodeCalls
1.3.6.1.4.1.3512.1.10.1.1.1.1

totalLatency
1.3.6.1.4.1.3512.1.10.1.1.1.2

opcodePlugInGroupIndex
1.3.6.1.4.1.3512.1.10.1.1.1.3

latencyTable
1.3.6.1.4.1.3512.1.10.1.2

latencyEntry
1.3.6.1.4.1.3512.1.10.1.2.1

```

latencyTableIndex
1.3.6.1.4.1.3512.1.10.1.2.1.1

latency
1.3.6.1.4.1.3512.1.10.1.2.1.2

latencyTimestamp
1.3.6.1.4.1.3512.1.10.1.2.1.3

Input Controller probes

ppl
1.3.6.1.4.1.3512.1.11

inputController
1.3.6.1.4.1.3512.1.11.1

inputControllerGroup
1.3.6.1.4.1.3512.1.11.1.1

inputControllerGroupEntry
1.3.6.1.4.1.3512.1.11.1.1.1

maxProcessingTime
1.3.6.1.4.1.3512.1.11.1.1.1.1

minProcessingTime
1.3.6.1.4.1.3512.1.11.1.1.1.2

inputControllerGroupIndex
1.3.6.1.4.1.3512.1.11.1.1.1.3

processingTimeTable
1.3.6.1.4.1.3512.1.11.1.2

processingTimeEntry
1.3.6.1.4.1.3512.1.11.1.2.1

processingTimeTableIndex
1.3.6.1.4.1.3512.1.11.1.2.1.1

processingTime
1.3.6.1.4.1.3512.1.11.1.2.1.2

timestamp
1.3.6.1.4.1.3512.1.11.1.2.1.3

Process Table

processTable
1.3.6.1.4.1.3512.1.101

processEntry
1.3.6.1.4.1.3512.1.101.1

processIndex
1.3.6.1.4.1.3512.1.101.1.1

processDescr
1.3.6.1.4.1.3512.1.101.1.2

Registry Table

registryTable
1.3.6.1.4.1.3512.1.102

registryEntry (1)
1.3.6.1.4.1.3512.1.102.1

registryIndex (1)
1.3.6.1.4.1.3512.1.102.1.1

registryName (2)
1.3.6.1.4.1.3512.1.102.1.2

Configuring Pipeline Manager

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

For background information, see "About the Pipeline Manager System Architecture" in *BRM Concepts*.

About Configuring Pipeline Manager

To configure Pipeline Manager, you use the following files:

- *Registry files*, which you use to configure a Pipeline Manager instance at system startup. See ["Using Registry Files to Configure Pipeline Manager"](#).
- *Semaphore files*, which you use to configure and control pipelines during runtime. See ["Using Semaphore Files to Control Pipeline Manager"](#).

You can also use the **pin_ctl** utility to start and stop Pipeline Manager. See ["Starting and Stopping the BRM System"](#).

Using Registry Files to Configure Pipeline Manager

A registry file is an ASCII text file that configures a Pipeline Manager instance at system startup. (There is one registry file for each Pipeline Manager instance.) You use a registry file to configure all of your Pipeline Manager system settings, such as the location of log files, your input stream format, data modules, pipelines, and the number of system threads.

Note: All directories and folders referenced in the registry file must exist before starting the Pipeline Manager.

After you have configured the registry file, you use the registry file name in the command for starting Pipeline Manager:

```
ifw -r RegistryFile
```

About the Registry File Structure

Registry files use a hierarchical structure, with each subsection nested within another. Each subsection provides the configuration for a module. These can be system modules, such as the Memory Monitor.

Each nested subsection is indented by several spaces and surrounded by curly braces { }. For example, the following shows how you specify the semaphore entries, **FilePath** and **FileName**:

```
ifw
{
...
    Semaphore
    {
        FilePath = /opt/ifw/semaphore
        FileName = semaphore.reg
    }
}
```

The registry hierarchy is shown in this chapter by the *dot* (.) convention. For example, this hierarchy:

```
ifw
{
...
    ProcessLog
    {
        Module
        {
            ...
        }
    }
}
```

is shown like this:

ifw.ProcessLog.Module

Where each period represents a level in the hierarchy.

The following shows the top-level subsections in the registry file. Each of these subsections controls a system-wide function as described in [Table 7-1](#). The **ifw.Pipelines** section contains system-wide entries that apply to all pipelines, and subsections for each pipeline.

```
ifw
{
    Instrumentation
    DiagnosticDataHandler
    ParallelLoadManager
    LogMessageTable
    Semaphore
    Registry
    ProcessLog
    MemoryMonitor
    EventHandler
    DataPool
    TransactionIdController
    SequencerPool
    Pipelines
    ...
}
```

where:

Table 7–1 Top-Level Subsections in Registry File

Registry entry	Description	Required
ifw	Specifies the registry name for the Pipeline Manager instance. This is always the first entry in the registry. It is read by the Pipeline Manager Controller. See "About the Controller" in <i>BRM Concepts</i> .	Yes
ifw.Instrumentation	Section that configures Operations Management Framework (OMF) instrumentation data collection. See "Enabling SNMP Instrumentation Data Collection" .	Yes
ifw.ParallelLoadManager	Section that configures multithreaded loading of your pipelines, data modules, and function modules. See "Reducing Startup Times with Parallel Loading" .	No
ifw.DiagnosticDataHandler	Section that configures diagnostic data collection. See "Using the Diagnostic Data Handler to Get OMF Diagnostic Data" .	Yes
ifw.LogMessageTable	Section that configures global log file setting. See "About Pipeline Manager Log Files" .	Yes
ifw.Semaphore	Section that defines the name and location of your semaphore files. See "Using Semaphore Files to Control Pipeline Manager" .	Yes
ifw.Registry	Section that defines the names and locations of the files that contain updated registry information. See "Controller" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Yes
ifw.ProcessLog	Section that configures your process log. For information about the log entries, see "About Pipeline Manager Log Files" .	Yes
ifw.MemoryMonitor	Section that configures memory monitoring. See "Monitoring Pipeline Manager Memory Usage" .	No
ifw.EventHandler	Section that configures the Event Handler. See "About the Event Handler" in <i>BRM Concepts</i> .	No
ifw.DataPool	Section that configures your data modules. See "Configuring the Data Pool" .	Yes
ifw.TransactionIdController	Section that configures your Transaction ID Controller. See "About the Transaction ID Controller" in <i>BRM Concepts</i> and "About Pipeline Manager Transactions" .	Yes
ifw.SequencerPool	Section that configures all Sequencers used by a single Pipeline Manager instance. See "Configuring Sequence Checking" .	No
ifw.Pipelines	Section that configures your individual pipelines. See "About Configuring Pipelines" .	Yes

About the Registry File Syntax

Most registry file entries are key-value pairs separated by an equal sign (=):

Entry = *Value*

where:

- *Entry* specifies the entry name. Ensure you use the correct entry name spelling and capitalization; entry names are case-sensitive.
- *Value* is a value specific to the configuration entry.

For example, **Source = File** or **Split = True**.

A few registry file entries, such as **Reload**, do not take a value. In these cases, follow the identifier with curly braces. For example, **Reload {}**.

Registry entries are either mandatory or optional. You must specify all mandatory entries. You can delete optional entries or comment them out by using a cross-hatch (#); Pipeline Manager uses default values for all unspecified optional entries.

Values can be either hard-coded and must be exact, or you can define your own.

- Hard-coded values must be entered exactly as documented. For example, when you enter a module name, it must have the correct spelling and capitalization. For FCT_Account, you cannot use FCT_account or FCT_ACCOUNT.
- Values that you define are often used elsewhere in the file, at which point they must be entered exactly as you defined them. For example, you might define the section for the DAT_AccountBatch module by using the entry **CustomerData**:

```
#-----  
# Infranet Customer Data  
#-----  
CustomerData  
{  
    ModuleName = DAT_AccountBatch
```

When you refer to that module elsewhere in the registry file, you point to **CustomerData**:

```
DataModule = ifw.DataPool.CustomerData
```

About the Sample Registry Files

Pipeline Manager includes the sample registry files listed in [Table 7–2](#) to help you get started:

Table 7–2 Pipeline Manager Sample Registry Files

File name	Directory location	Description
simple.reg	<i>Pipeline_home/samples/simple</i>	Simple registry file that you can use to verify that Pipeline Manager installed properly. It does not require a database. It tests the pipeline input, output, and runs one functional module (FCT_PrefixDesc).
wireless.reg	<i>Pipeline_home/conf</i>	Registry file that configures most function modules. This sample is a good place to start for creating your customized registry file.
wirelessRealtime.reg	<i>Pipeline_home/conf</i>	Registry file that configures most function modules for real-time features. This sample is a good place to start for creating your customized registry file.

About Configuring Pipelines

Pipelines perform the Pipeline Manager functions, such as rating and zoning. See "About Pipelines" in *BRM Concepts*.

You configure pipelines in the **ifw.Pipelines** registry section. For example, a Pipeline Manager configuration with multiple pipelines looks like this:

```
ifw  
{  
    ...  
    Pipelines  
    {  
        PipelineName
```

```

    {
        Input
        Functions
        Output
    }
    PipelineName
    {
        ...
    }
}
...

```

You can use any name you want to identify pipelines. You use that name in many places to point to the pipeline, so it should identify the function of the pipeline.

For each pipeline, you configure a pipeline controller. This section configures pipeline-specific configurations, such as threads, log files, the EDR Factory, and the **ifw.Pipelines.DataDescription** section. See the following topics:

- "About the Pipeline Controller" (see *BRM Concepts*)
- [Using Events to Start External Programs](#)
- [About Pipeline Manager Transactions](#)
- [About Pipeline Manager Log Files](#)

In addition, for each pipeline controller, you configure:

- An input section. See "Configuring EDR Input Processing" in *BRM Configuring Pipeline Rating and Discounting*.
- An **ifw.Pipelines.Functions** section. This section configures the function modules in the pipeline. The modules are run in the order that they are configured in this section. See ["About Configuring Function Modules"](#).
- An output section. See "Configuring EDR Output Processing" in *BRM Configuring Pipeline Rating and Discounting*.

The registry subsections in a pipeline are listed in [Table 7–3](#):

Table 7–3 Pipeline Registry Subsections

Registry Entry	Description	Required
ifw.Pipelines	Section that configures your individual pipelines.	Yes
ifw.Pipelines.PipelineName	Section that configures a single pipeline.	Yes
ifw.Pipelines.PipelineName.Input	Section that configures a pipeline's input module. For information about the input module entries, see "Pipeline Manager Input and Output Modules" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Yes
ifw.Pipelines.PipelineName.Functions	Section that configures a pipeline's function modules. For information about the function module entries, see "About Configuring Function Modules" .	Yes
ifw.Pipelines.PipelineName.Output	Section that configures a pipeline's output module. For information about the output module entries, see "Pipeline Manager Input and Output Modules" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Yes

About Configuring Function Modules

You configure function modules in the **ifw.Pipelines.Functions** section.

The **ifw.Pipelines.Functions** section uses this hierarchy:

```
ifw
{
...
  Pipelines
  {
    PipelineName
    {
      Input
      ...
      Functions
      {
        Function_pool_name
        {
          FunctionPool
          {
            Module_identifier
            {
              ModuleName = Module_executable
              Module
              {
                Entry = value
              }
            }
          }
        }
      }
    }
  }
...
}
```

The entries listed in [Table 7–4](#) are a combination of required text and text that you define.

Table 7–4 Pipeline Registry Functions Section Entries

Entry	Description
Functions	Section name. You must use Functions .
<i>Function_pool_name</i>	The name of the function pool. You define this name. See " Optimizing a Pipeline by Using Function Pools ".
FunctionPool	Section name. You must use FunctionPool .
<i>Module_identifier</i>	The descriptive module identifier. For example, the module identifier for FCT_Account in the sample registry is CustomerSearch . You define these names. They are often referenced by other modules; for example, to connect to the DAT_AccountBatch module, the FCT_Account module points to CustomerData .
ModuleName = Module_executable	ModuleName is the entry. You must use ModuleName . <i>Module_executable</i> is the name of the module; for example, FCT_Account. This name is case-sensitive and must be spelled correctly; for example, you must use FCT_Account, not FCT_account or FCT_ACCOUNT. You can find the exact spelling and capitalization by looking at the executable name in the <i>Pipeline_home/lib</i> directory.
Module	Section name. You must use Module .
<i>Entry = value</i>	These are the registry entries, for example: Active = True

This example shows a sample hierarchy. This sample does the following:

- Creates a function pool named *PreProcessing*.
- Runs the FCT_IRules module, using the identifier *PipelineSplit*.

```
Functions
{
    PreProcessing
    {
        FunctionPool
        {
            PipelineSplit
            {
                ModuleName = FCT_IRules
                Module
                {
                    Active = True
                }
            }
        }
    }
}
```

About iScripts and iRules

iScripts and iRules perform processing tasks similar to function modules. They are run by the FCT_iScript and FCT_iRules modules. In addition to the iScripts and iRules provided by BRM, you can create your own iScripts and iRules.

See "Creating iScripts and iRules" in *BRM Developer's Guide*.

About Configuring iScripts

To run iScripts, you use the FCT_iScript module. See "FCT_iScript" in *BRM Configuring Pipeline Rating and Discounting*.

The registry section for the FCT_iScript module includes the script to run, for example:

```
ApplyTaxIScript
{
    ModuleName = FCT_iScript
    Module
    {
        Active = True
        Source = File
        Scripts
        {
            ApplyTaxIScript
            {
                FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ApplyTax.isc
            }
        }
    }
}
```

You can provide registry parameters to use in the iScript. This example provides the iScript with a G/L code:

```
Scripts
{
    ConsolidatedCPIScript
    {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
        GL_CODE = 1514
    }
}
```

```
}
```

About Configuring iRules

To run iRules, you use the FCT_IRules modules. See "FCT_IRules" In *BRM Configuring Pipeline Rating and Discounting*.

To configure the FCT_IRules module, provide a connection to the Pipeline Manager database. The FCT_IRules module runs the rules that apply to the conditions in the pipeline. If a condition in a rule item matches the current EDR container, the evaluation stops and the script associated with the rule item is executed for the current EDR container.

This example shows a typical FCT_IRules registry section:

```
PipelineSplit
{
  ModuleName = FCT_IRules
  Module
  {
    Active = TRUE
    Source = Database
    DataConnection = integrate.DataPool.DataConnection
    Rules
    {
    }
  }
}
```

You can use the **Rules** entry to specify a specific script to run:

```
Rules
{
  TAP3_VAL
}
```

Configuring Multiple Instances of a Pipeline

To simplify the configuration of multiple pipelines, use the **ifw.Pipelines.Instances** subsection. Pipeline Manager reads the required number of instances for a given pipeline and instantiates each of them accordingly.

Note: The **ifw.Pipelines.Instances** subsection creates multiple instances of pipelines. To create multiple instances of sequencers, output streams, or system brands for multiple roaming partners, use the Instances module. See ["About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands"](#) for more information.

For example, this subsection configures ten instances of the authorization pipeline:

```
ifw
{
  ...
  Pipelines
  {
    Instances
    {
```



```
AuthPipeline
{
    NumberOfInstances = 10
    InstanceSpecificRegistries
    {
        Entry1 = TransactionManager.BinaryLogFileName
        Entry2 = PipelineLog.Module.ITO.FileName
        ...
    }
}
```

To specify instance-specific registry entries, you add the entries in the **ifw.Pipelines.Instances.Pipeline_Name.InstanceSpecificRegistries** section.

The pipeline generates the instance-specific log file names by adding the instance ID to the base pipeline file names.

For example, if the base pipeline file name for the TransactionManager log file is **binaryLogFile_RT_GPRS.dat**, then the instance-specific files generated are **binaryLogFile_RT_GPRS.dat0**, **binaryLogFile_RT_GPRS.dat1**, and **binaryLogFile_RT_GPRS.dat2**.

Note: If instance-specific entries are not specified, the pipeline uses the base pipeline configurations.

About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands

To manage multiple roaming partners, you can use the Instances module to configure multiple instances of sequencers, output streams, or system brands. You configure the Instances module by adding the **ifw.Instances** registry section in the roaming registry file (*Pipeline_home/conf/roaming.reg*).

Note: To create multiple instances of pipelines, use the **ifw.Pipelines.Instances** subsection. See ["Configuring Multiple Instances of a Pipeline"](#) for more information.

The Instances module configures multiple instances of sequencers, output streams, or system brands using template sections or entries in the roaming registry file. Instead of creating multiple sections of entries, you use the single section or entry templates in the roaming registry file. When the pipeline runs, data for each roaming partner is inserted into the templates, effectively instantiating multiple registry sections or entries. For example, if there are two roaming partners, OPRT1 and OPRT2, the template is instantiated into two sections of entries in the pipeline.

To identify which roaming partners to use with the template, the Instances module reads the roaming configuration data file generated by the **RoamingConfigGen64** utility. This file includes data for each of the roaming partners. For example, the data can include the sequencing information, output information, and so on.

You use the **SequencerPool** or **OUT_GenericStream** template section or the **SystemBrands** template entry in the roaming registry file to configure multiple sequencers, output streams, or system brands. These template sections or entries contain the variables that must be changed in each new instance of the **SequencerPool**

or **OUT_GenericStream** section or the **SystemBrands** entry instantiated in the pipeline.

The following example shows the **SequencerPool** template section:

```
SequencerPool
{
SEQ_GEN_TAPOUT_XXX
{
    Source = Database
    Controller
    {
        SequencerType = Generation
        ReuseGap = True
        SequenceLength = 5
        DatabaseConnection = ifw.DataPool.Login
    }
}
```

where **XXX** is the visiting network operator code that must be changed in each new instance of the **SequencerPool** section; for example, OPRT1, OPRT2, and so on.

Use the Instances module with the **RoamingConfigGen64** utility. The **RoamingConfigGen64** utility collects the roaming partner information from the Pipeline Manager database and creates the roaming configuration data file. The Instances module uses the values in the roaming configuration data file to replace the variables in each instance of the **SequencerPool** or **OUT_GenericStream** section or the **SystemBrands** entry instantiated in the pipeline.

When you run the **RoamingConfigGen64** utility, you specify a home network operator code. The utility searches the Pipeline Manager database to find the VPLMNs associated with that home network operator. For example, if the home network operator has two VPLMNs, a record for each of them is created in the roaming configuration data file.

The following example shows the roaming configuration data file generated by the **RoamingConfigGen64** utility:

```
# Column Headers
#####
#####
VPLMN|TAPOUT_SEQUENCER|NRTRDEOUT_SEQUENCER|TAPOUT_STREAM|NRTRDEOUT_STREAM|
TAPOUT_PATH|NRTRDEOUT_PATH|TAPOUT_PREFIX|NRTRDEOUT_PREFIX|TMP_PREFIX|
TMP_DATA_PREFIX#####
#####

OPRT1|SEQ_GEN_TAPOUT_OPRT1|SEQ_GEN_NRTRDEOUT_OPRT1|TAPOutput_OPRT1|NRTRDEOutput_
OPRT1|./data/outcollect/tapout/oprt1|./data/outcollect/nrtrdeout/oprt1|CDEUR01OPRT
1|NREUR01OPRT1|temptest_oprt1|temp.oprt1.tmp.|42|5|OUT_DevNull
OPRT2|SEQ_GEN_TAPOUT_OPRT2|SEQ_GEN_NRTRDEOUT_OPRT2|TAPOutput_OPRT2|NRTRDEOutput_
OPRT2|./data/outcollect/tapout/oprt2|./data/outcollect/nrtrdeout/oprt2|CDEUR01OPRT
2|NREUR01OPRT2|temptest_oprt2|temp.oprt2.tmp.|42|5|OUT_GenericStream
```

The following example shows the entries in the **ifw.Instances** registry section to configure multiple instances of sequencers:

```
{
    ifw
    {
        Instances
        {
            SEQ_GEN_TAPOUT
        }
    }
}
```

```

{
  BlockName = SequencerPool.SEQ_GEN_TAPOUT_XXX
  DataFile = ./RoamingPartnerConf.dat
  InstanceSpecificEntries
  {
    ModifyBlockName
    {
      Instance = [BlockName]
      UseColumn = TAPOUT_SEQUENCER
    }
  }
}

```

The following example shows the two instances of sequencers instantiated in the pipeline, based on the entries in the **ifw.Instances** registry section, using the TAPOUT_SEQUENCER values in the data file:

```

SequencerPool
{
  SEQ_GEN_TAPOUT_OPRT1
  {
    Source = Database
    Controller
    {
      SequencerType = Generation
      ReuseGap = True
      SequenceLength = 5
      DatabaseConnection = ifw.DataPool.Login
    }
  }
  SEQ_GEN_TAPOUT_OPRT2
  {
    Source = Database
    Controller
    {
      SequencerType = Generation
      ReuseGap = True
      SequenceLength = 5
      DatabaseConnection = ifw.DataPool.Login
    }
  }
}

```

See ["Configuring Multiple Instances of Sequencers, Output Streams, or System Brands"](#) for instructions.

Configuring Multiple Instances of Sequencers, Output Streams, or System Brands

To configure multiple instances of sequencers, output streams, or system brands:

1. Create the roaming configuration data file by running the following command:

```

RoamingConfigGen64 -l database_access_library -s server_name [-d database_name]
-c operator_code [-o output_path] [-b base_path]

```

where:

- *database_access_library* is the database access library; for example, **liboci10g6312d.a** for Oracle on AIX.
- *server_name* specifies the name of the host machine running the Pipeline Manager database.

- *database_name* specifies the database name of the Pipeline Manager database. The default is an empty string (' ').
- *operator_code* specifies the home network operator code. The default is **PORTL**.
- *output_path* specifies the output path for the data file generated by the **RoamingConfigGen64** utility. By default, the data file is saved in the *Pipeline_home/conf/* directory.
- *base_path* specifies the base path to the directory for Transferred Account Procedure (TAP) and Near Real Time Roaming Data Exchange (NRTRDE) output files. The default path is *Pipeline_home/data/outcollect/*

For example:

```
RoamingConfigGen64 -l liboci10g6312d.so -s $ORACLE_SID -d ' ' -c EUR01  
-o Pipeline_home/conf/ -b Pipeline_home/data/outcollect/
```

For more information about the **RoamingConfigGen64** Perl script, see "RoamingConfigGen64" in *BRM Configuring Pipeline Rating and Discounting*.

2. Open the roaming registry file (*Pipeline_home/conf/roaming.reg*) file in a text editor.
3. Ensure that the **SequencerPool** or **OUT_GenericStream** template section or the **SystemBrands** template entry exists in the roaming registry file.

If the template for the roaming registry section or entry you want to instantiate does not exist, create a template for that registry section or entry in the file.

The following example shows the **SequencerPool** template section:

```
SequencerPool  
{  
  SEQ_GEN_TAPOUT_XXX  
  {  
    Source = Database  
    Controller  
    {  
      SequencerType = Generation  
      ReuseGap = True  
      SequenceLength = 5  
      DatabaseConnection = ifw.DataPool.Login  
    }  
  }  
}
```

4. Add the instance-specific entries in the **ifw.Instances.InstantiationName.InstanceSpecificEntries** subsection. If the **ifw.Instances** registry section does not exist, you must add the section in the file.

The **ifw.Instances** registry section uses the following hierarchy:

```
Instances  
{  
  InstantiationName  
  {  
    BlockName = TemplatePath  
    DataFile = DataFilePath  
    InstanceSpecificEntries  
    {  
      InstanceChangeName  
      {  
        Instance = InstanceValue  
        UseColumn = ColumnName  
        Mode = ModeValue  
      }  
    }  
  }  
}
```

```

    }
  }
}

```

where:

- *InstantiationName* is the descriptive name of the instantiation; for example, **SEQ_GEN_TAPOUT**.
- *TemplatePath* is the template section or entry in the roaming registry file that is used to instantiate multiple registry sections or entries. For example, **SequencerPool.SEQ_GEN_TAPOUT_XXX**
- *DataFilePath* is the path to the data file generated by the **RoamingConfigGen64** utility; for example, *Pipeline_home/conf/RoamingPartnerConf.dat*.
- *InstanceChangeName* is the descriptive name of the change required in each instance; for example, **ModifyBlockName**.
- *InstanceValue* specifies whether to change the section name, entry name, or the value of the entry in each new instance created.

The valid values are:

- **[BlockName]** specifies that the section name or entry name must be changed in each new instance.
- **[BlockValue]** specifies that the value of the entry must be changed in each new instance.
- *RegistryEntry* specifies the entry in the template section for which the value must be changed in each new instance; for example, **Module.Recipient**.
- *ColumnName* is the column in the data file generated by the **RoamingConfigGen64** utility that is used to change the section name, entry name, or the value of the entry in each instance according to the change mode. For example, **TAPOUT_SEQUENCER**.
- *ModeValue* is the mode of changing (such as **REPLACE**) the section name, entry name, or the value of the entry in each instance using the column values in the data file generated by the **RoamingConfigGen64** utility.

For more information on the Instances module, see "Instances" in *BRM Configuring Pipeline Rating and Discounting*.

5. Save and close the file.
6. Stop and restart Pipeline Manager.

Configuring the Data Pool

To configure data modules, you configure the **ifw.DataPool** registry subsection. This subsection uses the following hierarchy:

```

DataPool
{
  Module_identifier
  {
    ModuleName = Module_executable
    Module
    {
      Entry = value
    }
  }
}

```

The entries listed in [Table 7–5](#) are a combination of required text and text that you define.

Table 7–5 Pipeline Registry DataPool Section Entries

Entry	Description
DataPool	Section name. You must use DataPool .
<i>Module_identifier</i>	<p>The descriptive module identifier. For example, in the sample registry, the module identifier for DAT_AccountBatch is CustomerData.</p> <p>You define these names. They are often referenced by other modules; for example, to connect to the DAT_AccountBatch module, the FCT_Account module points to CustomerData.</p>
ModuleName = <i>Module_executable</i>	<p>ModuleName is the entry. You must use ModuleName.</p> <p><i>Module_executable</i> is the name of the module; for example, DAT_AccountBatch. This name is case-sensitive and must be spelled correctly; for example, you must use DAT_AccountBatch, not DAT_Accountbatch or DAT_Account_Batch.</p> <p>You can find the exact spelling and capitalization by looking at the executable name in the <i>Pipeline_home/lib</i> directory.</p>
Module	Section name. You must use Module .
<i>Entry = value</i>	<p>These are the registry entries; for example:</p> <p>Active = True</p>

This example shows a sample hierarchy:

```
DataPool
{
    CustomerData
    {
        ModuleName = DAT_AccountBatch
        Module
        {
            IntegrateConnection = ifw.DataPool.Login
```

Connecting a Module to a Database

You connect modules to the Pipeline Manager database and the BRM database through the Database Connect module. To do so:

1. Configure the Database Connect module in the **ifw.DataPool** section of the registry file. For information, see "Database Connect (DBC)" in *BRM Configuring Pipeline Rating and Discounting*.

You can configure three types of connections:

- A connection to the Pipeline Manager database.
 - A connection to the BRM database.
 - A connection to the database login queue (used by the DAT_Listener module).
2. When configuring a module that needs a connection to the Pipeline Manager database, use one of the following registry entries:

- **DataConnection**
- **IntegrateConnection**

These entries do the same thing; they point to the **ifw.DataPool.Login** section. For example:

```
DataConnection = ifw.DataPool.Login
IntegrateConnection = ifw.DataPool.Login
```

See the documentation for each module to determine which entry to use.

Note: Some modules can get data either from the database or from a file. If you configure the module to get data from a file, the module does not connect to the database.

3. When configuring a module that needs a connection to the BRM database, configure one of the following registry entries:

- **DataConnection**
- **InfranetConnection**

These entries do the same thing; they point to the **ifw.DataPool.LoginInfranet** section. For example:

```
DataConnection = ifw.DataPool.LoginInfranet
InfranetConnection = ifw.DataPool.LoginInfranet
```

Forcing a Database Reconnection

You can force the Database Connect module to reconnect to the Pipeline Manager database by using the following semaphore entry:

```
ifw.DataPool.Login.Module.Reconnect {}
```

This semaphore closes all open database connections and reconnects the Database Connect module to the Pipeline Manager database.

For information on how to create semaphore files, see ["Updating Configuration Settings during Runtime by Using Semaphore Files"](#).

Reloading Data into a Pipeline Manager Module

When you update data in the Pipeline Manager database, it is not automatically loaded into the modules. For example, if you change pricing data, EDRs continue to be rated by using the old pricing data until the new data is loaded into the data modules.

You use the **Reload** semaphore entry to reload data from the database into a module.

If the reload operation does not succeed, the module stops processing EDRs until data is loaded correctly. In some cases, you can configure how a module behaves if reloading fails:

- To configure a module to immediately resume processing using the previous data, set its **ReuseOnFailure** startup registry entry to **True**. Not all modules have this registry entry. Check the module's reference documentation to determine whether its registry includes **ReuseOnFailure**.

- To ensure that a module does not resume processing EDRs until the latest data is loaded, do not include **ReuseOnFailure** in the registry. This is the only option for modules that do not include this registry entry.

Using Business Parameter Settings from the BRM Database

You enable or disable optional BRM features and functionality by configuring business parameter settings, which are stored in `/config/business_params` objects in the BRM database. Pipeline Manager can determine whether these features and functionality are enabled by using the `DAT_PortalConfig` module, which retrieves and stores business parameter settings from the BRM database at pipeline initialization. Any other data modules that need a business parameter setting retrieve it directly from the `DAT_PortalConfig` module's internal memory.

Table 7–6 lists the data modules that use business parameter settings, the features that depend on the setting, and the `/config/business_params` parameter class and entry that each feature uses:

Table 7–6 Data Modules Using Business Parameter Settings

Pipeline Manager module	Feature	Parameter class	/config/business_params entry
DAT_AccountBatch	Balance monitoring. See "About Balance Monitoring" in <i>BRM Managing Accounts Receivable</i> .	multi_bal	BalanceMonitoring
DAT_BalanceBatch	Validity end time for first-usage resources. See "About Restricting the End Time of Granted Resources that Start on First Usage" in <i>BRM Setting Up Pricing and Rating</i> .	multi_bal	RestrictResourceValidityToOffer SortValidityBy CreditThresholdChecking
DAT_Discount	Discount validity and exclusion rules. See "About Discount Exclusion Rules" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	billing	ValidateDiscountDependency
DAT_PriceModel	Pricing model for pipeline rating. See "About Pipeline Rating" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	subscription	TailormadeProductsSearch
DAT_RatePlan	Rate plan for pipeline rating. See "About Configuring Pipeline Rating" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	subscription	TailormadeProductsSearch

To set up Pipeline Manager to use business parameter settings from the BRM database, perform these tasks:

1. Configure the `DAT_PortalConfig` module in your registry file. This module must be listed in the registry file before any other data modules that are connected to it. See "DAT_PortalConfig" in *BRM Configuring Pipeline Rating and Discounting*.
2. Configure data modules to retrieve business parameter settings from `DAT_PortalConfig`. See ["Connecting Pipeline Manager Modules to DAT_PortalConfig"](#).

After Pipeline Manager starts, you can:

- Verify that the entries loaded properly by printing the parameters that DAT_PortalConfig has stored in memory. See ["Printing Business Parameter Settings Stored in DAT_PortalConfig Memory"](#).
- Refresh business parameter settings stored in the DAT_PortalConfig module's internal memory. See ["Refreshing Business Parameter Settings Stored in DAT_PortalConfig Memory"](#).

Connecting Pipeline Manager Modules to DAT_PortalConfig

You must connect all data modules in your system that need business parameter settings to DAT_PortalConfig. You connect a module to DAT_PortalConfig by using the module's **PortalConfigDataModule** registry entry. For example:

```
PortalConfigDataModule=ifw.DataPool.PortalConfigDataModule
```

Note: You can use any name you want to identify the registry section that configures DAT_PortalConfig, but you must use that name exactly when configuring modules to point to that registry section.

For example, the following entry, shown in bold, connects the DAT_Discount module to DAT_PortalConfig:

```
#-----
# Discount Model Data Module
#-----
DiscountModelDataModule
{
  ModuleName = DAT_Discount
  Module
  {
    InfranetConnection      = ifw.DataPool.LoginInfranet
    IntegrateConnection     = ifw.DataPool.Login
    PortalConfigDataModule = ifw.DataPool.PortalConfigDataModule
    AccountDataModule       = ifw.DataPool.CustomerData
  }
}
```

Printing Business Parameter Settings Stored in DAT_PortalConfig Memory

To print to a file the business parameter settings stored in the DAT_PortalConfig module's memory, use the CBPPrintData semaphore (see "DAT_PortalConfig" in *BRM Configuring Pipeline Rating and Discounting*). For example:

```
ifw.DataPool.PortalConfig.Module.CBPPrintData=[Path] [Filename]
```

where:

- *Path* specifies where to create the output file. By default, the file is created in the current directory.
- *Filename* specifies the name for the output file. The default file name is **DefaultCBPDDataFile_timestamp.lst**. The module appends a timestamp to the end of the file name to prevent the module from overwriting existing files.

For example:

```
ifw.DataPool.PortalConfig.Module.CBPPrintData=Portal/text/prntdata
```

When you submit the print semaphore, DAT_PortalConfig generates an output file that uses the format shown below:

```
<BusParamConfiguration>
  <BusParamConfigurationList>
    <ParamClass name="group_name">
      <Param>
        <Name>parameter_name</Name>
        <Type>data_type</Type>
        <Value>parameter_value</Value>
      </Param>
    </ParamClass>
  </BusParamConfigurationList>
</BusParamConfiguration>
```

For example, the following shows a sample output file for the **billing** parameter class:

```
<BusParamConfiguration>
  <BusParamConfigurationList>
    <ParamClass name="billing">
      <Param>
        <Name>rerate_during_billing</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
      <Param>
        <Name>validate_discount_dependency</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
      <Param>
        <Name>sub_bal_validity</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
    </ParamClass>
  </BusParamConfigurationList>
</BusParamConfiguration>
```

For information about semaphores, see ["Using Semaphore Files to Control Pipeline Manager"](#).

Refreshing Business Parameter Settings Stored in DAT_PortalConfig Memory

You must refresh DAT_PortalConfig memory whenever you update the **BalanceMonitoring**, **RestrictResourceValidityToOffer**, or **ValidateDiscountDependency** business parameter settings in the BRM database.

You refresh the memory by using the CBPREload semaphore entry (see "DAT_PortalConfig" in *BRM Configuring Pipeline Rating and Discounting*). For example:

```
ifw.DataPool.PortalConfigDataModule.Module.CBPREload{}
```

For information about semaphores, see ["Using Semaphore Files to Control Pipeline Manager"](#).

Connecting a Pipeline Manager Module to Another Module

Most function modules connect to data modules to get configuration data. For example, the FCT_Account module requires a connection to the DAT_AccountBatch module. Also, some data modules connect to other data modules.

To connect one module to another, you configure a registry entry for the module that requires the connection. For example, to connect the FCT_Account module to the DAT_AccountBatch module, you enter this when you configure the FCT_Account module:

```
DataModule = ifw.DataPool.CustomerData
```

CustomerData identifies the DAT_AccountBatch module, which is configured in the registry like this:

```
#-----
# Infranet Customer Data
#-----
CustomerData
{
    ModuleName = DAT_AccountBatch
```

Note: You can use any name you want to identify the registry section that configures a module, but you must use that name exactly when configuring modules to point to that registry section.

A function module can connect to more than one data module. For example, the FCT_ApplyBalance module includes two data module connection entries:

```
DiscountDataModule = ifw.DataPool.DiscountModelDataModule
BalanceDataModule = ifw.DataPool.BalanceDataModule
```

In addition, function modules, like data modules, can require a connection to the Pipeline Manager or BRM database, for example:

```
DataConnection = ifw.DataPool.LoginInfranet
```

Configuring Pipeline Buffers

Pipeline Manager uses buffers to control the flow of data moving from one thread to another. For example, you insert a buffer block into the LOG module to temporarily store log data received from your thread before it is written by the logging thread to a file.

To insert a buffer, you configure the pipeline's or module's **Buffer**, **InputBuffer**, or **OutputBuffer** registry section. In each section, you specify the buffer's type and size. Pipeline Manager supports the following buffer types:

- Rogue Wave buffers. See ["Using Rogue Wave Buffers"](#).
- Block transfer buffers. See ["Using Block Transfer Buffers on Solaris Systems"](#).
- Array buffers. See ["Using Array Buffers on HP-UX Itanium and Solaris Systems"](#).

Important: When configuring buffers in multiple function pools, each buffer must have a unique name.

Using Rogue Wave Buffers

By default, all buffers in Pipeline Manager are Rogue Wave buffers. These buffers are simple FIFO buffers of a configurable size. When a thread writes to or reads from a Rogue Wave buffer, it locks the entire buffer to ensure the integrity of the data. For example, if a Rogue Wave buffer has 15 containers, all 15 containers are locked when a thread accesses the buffer. Other threads must wait for the buffer to be unlocked before they can read or write data. For this reason, Rogue Wave buffers work best when only one thread will access the buffer.

Note: If multiple threads will access the buffer, use a block transfer. See "[Using Block Transfer Buffers on Solaris Systems](#)".

When a thread attempts to write to a full buffer or read from an empty buffer, the thread sleeps before attempting to access the buffer again.

To use a Rogue Wave buffer, you specify only the size of the buffer, by using the **Size** registry entry. This entry, listed in [Table 7-7](#), goes in the **Buffer**, **InputBuffer**, or **OutputBuffer** registry section.

Table 7-7 *Rogue Wave Buffers Registry Entry*

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes

The following shows sample registry entries for a Rogue Wave buffer:

```
Buffer
{
    Size = 100
}
```

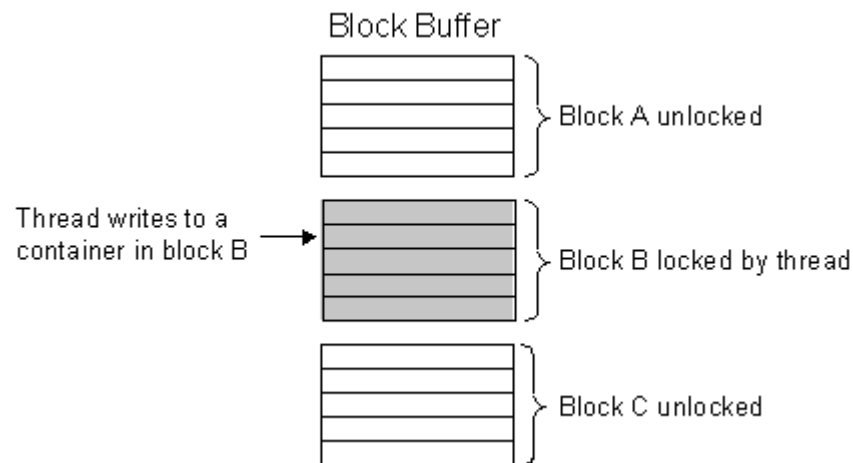
This registry example creates a Rogue Wave buffer with 100 containers.

Using Block Transfer Buffers on Solaris Systems

Block transfer buffers address performance and scalability issues that occur when two or more threads are accessing the same buffer. They are recommended for use on Solaris systems.

A block transfer buffer is a buffer that is separated into logical subsections (or blocks) with a configurable number of buffer containers. When a thread accesses a buffer container, it locks only those containers that are in the same block. This enables other threads to access other buffer containers in the remaining free blocks.

For example, a buffer has 15 containers separated into 3 logical blocks. A thread writing to a container in block B locks the block to prevent other threads from changing the container's value during the write operation. Threads dedicated to blocks A and C can still read and write data because those blocks are unlocked as shown in [Figure 7-1](#).

Figure 7-1 Block Transfer Buffer Locking

When a thread attempts to write to a full buffer or read from an empty buffer, the thread sleeps before attempting to access the buffer again.

To use a block transfer buffer, you use the buffer registry entries listed in [Table 7-8](#). These entries go in the **Buffer**, **InputBuffer**, or **OutputBuffer** section.

Table 7-8 Solaris Block Transfer Buffers Registry Entries

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes
BlockTransfer	Specifies whether the buffer operates in block transfer mode. True: The buffer operates in block transfer mode. False: The buffer does not operate in block transfer mode. If set to False , the pipeline ignores the BlockSize registry entry. The default is False .	Yes
BlockSize	Specifies the size of each buffer block.	Yes, if BlockSize is set to True .

The following shows sample registry entries for a block transfer buffer:

```
Buffer
{
    Size = 4000
    BlockTransfer = True
    BlockSize = 500
}
```

This example specifies a buffer size of 4,000 containers and a block size of 500 containers. Therefore, the buffer has eight (4,000/500) blocks.

Using Array Buffers on HP-UX Itanium and Solaris Systems

Array buffers address performance and scalability issues that occur when two or more threads are accessing the same buffer. They are recommended for use on HP-UX Itanium and Solaris systems.

Array buffers are similar to Rogue Wave buffers, except threads never lock the buffer. To protect shared data, threads use a compare and swap (CAS) method that atomically compares a container's old and current values before writing new data. This enables a

thread to read data from a buffer container, modify it, and write it back only if no other thread modified it in the meantime.

When writing data to an array buffer, a thread performs the following:

1. Reads the buffer container. The thread takes three arguments: the container's address, the container's current value, and the new value.
2. Updates the local variable with the new value.
3. Determines whether any other thread modified the container in the interim by comparing the container's current value with the value it initially read in step 1:
 - If the value *has not* changed, the thread writes the new value to the container.
 - If the value *has* changed, another thread modified the container during the write operation. The thread does not change the value and instead starts the process over at step 1.

When a thread attempts to write to a full buffer or read from an empty buffer, the thread spins for a specified number of times and then enables another thread to access the buffer before spinning again. You can specify the maximum number of times the thread yields before sleeping. The thread sleeps for a specified amount of time before starting the spin-and-yield process again.

For example, if the maximum number of spins is 2, the maximum number of yields is 2, and the sleep time is 10 milliseconds, the thread performs the following while waiting for a buffer container to become available:

1. Spins 2 times.
2. Yields to another thread.
3. Spins 2 times.
4. Yields to another thread.
5. Sleeps for 10 milliseconds.
6. Starts over at step 1.

To use an array buffer, you use the buffer registry entries listed in [Table 7–9](#). These entries go in the **Buffer**, **InputBuffer**, or **OutputBuffer** section.

Table 7–9 Solaris Array Buffers Registry Entries

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes
ArrayType	Specifies whether the buffer is an array buffer. True: The buffer is an array buffer. False: The buffer is not an array buffer. If set to False , the pipeline ignores the SpinCount , YieldCount , and SleepTimeMilliSec registry entries. The default is False .	Yes
SpinCount	Specifies the maximum number of times the thread spins while waiting for a buffer container to become available.	No
YieldCount	Specifies the maximum number of times a thread yields to another thread before the thread starts a sleep cycle.	No
SleepTimeMilliSec	Specifies how long the thread sleeps, in milliseconds, before trying to access the buffer again.	No

The following shows sample registry entries for an array buffer:

```
Buffer
{
  Size = 100
  ArrayType = True
  SpinCount = 100
  YieldCount = 100
  SleepTimeMilliSec = 10
}
```

Using Semaphore Files to Control Pipeline Manager

You use semaphore files to configure and control Pipeline Manager during runtime. They enable you to perform business tasks regularly without having to stop and restart the pipeline. For example, you can use semaphore files to stop a module or to reload data from the database.

The Controller checks for new semaphore files to process at a regular interval. You configure where and how often the Controller checks for new semaphore files by using the **Semaphore** and **ProcessLoopTimeout** registry entries. See "Controller" in *BRM Configuring Pipeline Rating and Discounting*.

When the Controller finds a semaphore file, it:

1. Prevents new transactions from being created.
2. Finishes processing all open transactions in the framework.
3. Stops the pipeline framework.
4. Loads the semaphore file into memory.
5. Changes the specified configuration settings and/or executes the specified semaphores.
6. Logs any processing errors in the **process.log** file.
7. Renames or deletes the semaphore file from the directory.

You configure the Controller to rename or delete semaphore files by using the **RetainFiles** semaphore entry.

8. Stops and restarts the pipeline framework.

For information on creating semaphore files, see ["Updating Configuration Settings during Runtime by Using Semaphore Files"](#).

Updating Configuration Settings during Runtime by Using Semaphore Files

To change the Pipeline Manager configuration during runtime, you must:

1. Specify where and how often the Controller checks for semaphore files. See ["Configuring Where and How Often the Controller Checks for Semaphore Files"](#).

Note: You perform this procedure only once, when you first configure your registry file.

2. Create your semaphore files. See ["Procedure for Updating Configuration Settings"](#).

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to for system administration tasks. See ["Using Perl Scripts to Administer"](#)

Pipeline Manager".

Configuring Where and How Often the Controller Checks for Semaphore Files

You use the following registry entries in [Table 7–10](#) to specify where and how often the Controller checks for semaphore files:

Table 7–10 Controller Configuration Registry Entries

Semaphore	Value	Description	Mandatory
ifw.ProcessLoopTimeout	Integer	Specifies the interval, in seconds, between polling for a new semaphore file. Note: This parameter controls the overall event loop, which includes looking for semaphore files.	Yes
ifw.Semaphore.FilePath	String	Specifies the directory where the Controller checks for semaphore files.	Yes
ifw.Semaphore.FileName	String	Specifies the name of the semaphore file.	Yes
ifw.Semaphore.RetainFiles	True False	Specifies whether semaphore files are deleted or saved after they are processed. <ul style="list-style-type: none">▪ True specifies to save semaphore files. The Controller renames the file by appending the current timestamp to the file name in the format <code>YYYYMMDD_hhmmss</code> and logs the semaphore file's new name in the process.log file. For example, the semaphore.reg file is renamed semaphore.reg_20031022_120803.▪ False specifies to delete semaphore files immediately after they are processed. The default is False .	No

Sample Registry Entries

```
ifw
{
    ...
    ProcessLoopTimeout = 30
    ...
    Semaphore
    {
        FilePath = /opt/ifw/semaphores
        FileName = semaphore.reg
        RetainFiles = True
    }
    ...
}
```

Procedure for Updating Configuration Settings

To update configuration settings during runtime:

1. Create a semaphore file using the file name specified in the registry file. (The examples in this chapter use **semaphore.reg**.)
2. Add new configuration or semaphore entries to the file. See ["Semaphore File Syntax"](#).

Note: The maximum number of entries you can add is **10000**.

3. Copy the semaphore file to the semaphore directory.

Important:

- Some settings in the registry file cannot be configured by using semaphore files. For a list of commands that can be submitted by using semaphores for a particular module, see the *Semaphore file entries* section in the documentation for the module.
 - Before you submit a semaphore to Pipeline Manager, be sure that Pipeline Manager has finished starting up. (It displays the message **Ready for processing**.) If a semaphore file is submitted when Pipeline Manager is still starting, the system renames the semaphore file, logs a message that the semaphore file was renamed, and ignores the renamed file. The file is left in the semaphore input directory. To execute the semaphore after the system completes startup, rename the file manually.
 - If a pipeline fails to process an update semaphore, the pipeline stops. To start it again, you must send another semaphore. See ["Starting and Stopping Individual Pipelines"](#).
-
-

Semaphore File Syntax

Semaphore commands use one of these formats:

- Key-value pair format, such as **LoadZoneDescription = True**. These semaphore commands require a value.

Note: The semaphore command fails if you do not supply a value.

- Semaphore entry { } format, such as **Reload{}**.

The commands in the semaphore file can be expressed in a nested hierarchy format or in a flattened syntax that uses periods to delimit nested sections. The syntax of a command reflects the hierarchical structure of the registry.

Important: You must specify the full path for the command when using either the hierarchy or the flattened format.

The following examples show how to set the process log file name by using the hierarchy and flattened formats.

Hierarchy Format

```
ifw
{
    ProcessLog
    {
        Module
        {
            ITO
            {
                FileName = process
            }
        }
    }
}
```

```
}  
}
```

Flattened Format

```
ifw.ProcessLog.Module.ITO.FileName = process
```

Though registry files can vary in structure, commands for each type of module follow a similar pattern. For function modules, the syntax follows this pattern (shown in flattened format):

```
ifw.Pipelines.Pipeline_Name.Functions.Function_pool_name.  
FunctionPool.Module_identifier.Module.Entry = Value
```

For example:

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.  
Aggregate.Module.Active = False
```

For data modules, the syntax is:

```
ifw.DataPool.Module_identifier.Module.Entry = Value
```

For example:

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.  
ZM_MOBILE = /data9/INTEGRATE/test/config/ZM_MOBILE-new.dat
```

You can specify multiple commands in one semaphore file by placing each command on a separate line. For example:

```
ifw.Pipelines.ALL_RATE.Active = True  
ifw.ProcessLog.Module.ITO.FileName = process
```

Important: Avoid using multi-command semaphore files unless you are sure that each command works without error when submitted in a single-command semaphore file. For more information, see ["Semaphore Error Messages"](#).

Semaphore Error Messages

When a semaphore command is executed correctly, the registry entry is removed and a success message is written to the process log.

If no command in a semaphore file can be executed correctly, the warning message **Semaphore was not processed; check spelling** is written to the process log.

Note: When processing a multi-command semaphore file, if at least one command in the file runs successfully, the pipeline does not log a message indicating that a command has failed.

For more information on the process log, see ["About Pipeline Manager Log Files"](#).

Using Events to Start External Programs

To use pipeline events to trigger external programs, use the Event Handler. See "About the Event Handler" in *BRM Concepts*.

Note:

- See the module reference documentation to find the events that a module sends. For example, to find the events that the DAT_ExchangeRate module uses, see "DAT_ExchangeRate" in *BRM Configuring Pipeline Rating and Discounting*.

Events are named like this:

- EVT_RELOAD_SUCCESSFUL
- EVT_RELOAD_FAILED
- You can configure modules to send custom events to the Event Handler by using iScripts. For information, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

About Mapping Events to Programs

You map events to programs by using the registry file's **Event** subsection.

The **Events** subsection specifies the module and event combinations can trigger an external program. Use the following syntax to create the Events subsection:

```
Events
{
    ModuleSendingEvent
    {
        EventName = Action
        EventName = Action
        TimeToWait = WaitValue
    }
}
```

where:

- *ModuleSendingEvent* specifies the registry name of the module that sends the event to the Event Handler. Add an entry for each module that can trigger an external program.

You can use wild cards (*) to specify multiple modules. For example, use **ifw.Pipelines.*** to specify all modules nested under the **ifw.Pipelines** section of the registry file.

- *EventName* specifies the event that triggers an external program. Add an entry for each event that triggers an external program.
- *Action* specifies the external program that is triggered by the event. Specify both the path and file name of the script or program.
- *WaitValue* specifies the time in seconds that the Event Handler waits for the external program to terminate. See "[Controlling External Programs](#)".

For example:

```
Events
{
    ifw.DataPool.Customer.Module
    {
        EVT_ReloadSuccess = ./script/script_1
        EVT_ReloadFailed = ./script/script_2
        TimeToWait = 30
    }
}
```

Note: You cannot change this event-to-program mapping while Pipeline Manager is running. To map an event to a new script or change the existing mapping, you must edit the registry file and stop and restart Pipeline Manager.

Controlling External Programs

Use the **TimeToWait** registry entry to specify the time in seconds that the Event Handler waits for the external program to terminate. If the program does not terminate before the **TimeToWait** period ends, the external program is killed.

If an event is received while an external program is running, the event is queued and is started after the running program terminates.

When this option is specified, only one external program can be run at a time.

If **TimeToWait** is not enabled, the EventHandler does not wait for the external program to finish its job. Instead it starts new external programs depending on the events in the queue.

By default, no **TimeToWait** value is assumed.

About Running External Programs

The Event Handler can run only one external program at a time. If the Event Handler receives an event while an external program is running, it queues the event until the program terminates.

Troubleshooting Event Handling

You can log the events that a data module receives. This enables you to test event logging. To do so, set the data module's **LogEvents** registry entry to **True**. By default, event logging is off.

Note: Not all data module support event logging. See the documentation for the data module that you are configuring.

About Pipeline Manager Transactions

Pipeline Manager uses transactional processing to ensure data integrity. When a system crash or power outage occurs, Pipeline Manager performs an automatic rollback and continues processing. Usually, the last CDR file that was being processed is rolled back and processed again.

In some cases, Pipeline Manager recognizes an inconsistent state of the file system; for example, an output file is missing. In these cases, Pipeline Manager does not restart and gives an error message.

Note: A transaction can consist of one CDR file or multiple CDR files. You define the number of CDR files in a transaction by configuring the **UnitsPerTransaction** entry. For information, see ["Combining Multiple CDR Files into One Transaction"](#).

Pipeline Manager uses two components for transaction handling:

- The Transaction Manager handles transactions for a single pipeline. See ["About the Transaction Manager"](#).
- The Transaction ID Controller manages transaction IDs for the entire Pipeline Manager instance. See ["Configuring the Transaction ID Controller"](#).

About the Transaction Manager

The Transaction Manager is a mandatory pipeline component that coordinates the state of all transactions in one pipeline.

The Transaction Manager performs the following functions:

- Monitors a transaction's state. Transactions move through these three states:
 - Opened (started)
 - Prepared
 - Closed (ended)
- Persists state information to the binary log file. For information, see ["About Transaction Log Files"](#).

When a transaction is in progress, the following occurs:

1. The Input Controller notifies the Transaction Manager that a transaction started.
2. The Transaction Manager requests a transaction ID number from the Transaction ID Controller. See ["Configuring the Transaction ID Controller"](#).
3. The Transaction ID Controller issues the next ID number to the Transaction Manager.
4. The Input Controller, function modules, and Output Controller process the input stream and notify the Transaction Manager if any of the following are required:
 - Rollback. If a rollback is required, the Transaction Manager rolls back the transaction and undoes all changes.

Note: When redo is enabled, the Transaction Manager also cancels any newly opened transactions.

- Cancel. If a cancel is required, the Transaction Manager undoes all changes made during the transaction.
5. The Output Controller notifies the Transaction Manager that the transaction ended.
 6. The Transaction Manager requests the Input Controller, function modules, and Output Controller to prepare for a commit of the transaction.
 7. The Transaction Manager performs one of the following:
 - If all of the modules prepare successfully, the Transaction Manager commits the transaction.
 - If the prepare fails, the Transaction Manager rolls back the transaction.

Two special types of EDRs are used for managing transactions:

- Before EDRs are processed, a *begin transaction EDR* is created. This tells Pipeline Manager which EDRs are part of the transaction.
- After all EDRs are processed, an *end transaction EDR* is created. When this EDR arrives at the output, the transaction can be committed.

You configure your Transaction Managers by using the **TransactionManager** section of the registry file. For information, see "Transaction Manager" in *BRM Configuring Pipeline Rating and Discounting*.

About Cancelling Transactions When a Rollback Occurs

Use the Transaction Manager **RedoEnabled** registry entry to cancel all open transactions if a rollback occurs.

When a rollback is demanded, the Transaction Manager performs the following:

1. Disables the creation of new transactions.
2. Rolls back all attached modules.
3. Cancels any open transactions.
4. Re-enables the creation of new transactions.

When **RedoEnabled** is disabled, the Transaction Manager only rolls back the attached modules.

About Transaction Log Files

All dynamic data, for example, aggregation results, call assembling records, and duplicate check data, is always kept in main memory. In addition, to ensure transactional integrity, data in memory has to be made persistent. To do so, transactional modules write data to work files. Data in the work files is used to record the status of the transaction.

Each Transaction Manager generates its own binary log file, which stores information about a pipeline's currently open transactions. The Transaction Manager writes information to the file when a transaction starts or changes state and deletes the transaction from the file when it ends. Thus, the file's size changes constantly.

The binary log file stores the following for each open transaction:

- The transaction's starting timestamp.
- Transaction ID number.
- The list of CDR files that make up the transaction.
- Whether any of the following occurred:
 - Rollback
 - Cancel
 - Redo
 - Prepare

You should regularly back up binary log files. These files are needed when you stop and restart Pipeline Manager to resolve any open transactions at the time of failure.

Note: When you stop and restart Pipeline Manager after an ungraceful shutdown, the Transaction Manager commits all prepared transactions and rolls back all other uncommitted transactions.

Configuring the Transaction ID Controller

You configure the Transaction ID Controller by using the **ifw.TransactionIDController** section of the registry file. For information, see "About the Transaction ID Controller" in *BRM Concepts* and "Transaction ID Controller" in *BRM Configuring Pipeline Rating and Discounting*.

About Storing IDs in Cache

When the Transaction ID Controller must cache a block of IDs, it does the following:

1. Accesses the state file or table for the increment value and last issued ID number.
2. Caches the next block of transaction IDs.

For example, if the last ID is 200 and the increment value is 100, the Transaction ID Controller caches IDs 201 through 300.

3. Resets the last ID number in the state table or file.

In the example above, the Transaction ID Controller sets the last ID to 300.

You configure the number of IDs stored in cache by using the Increment registry entry.

About the Transaction ID State File and Table

The state file or table stores the last issued transaction ID number and the configured increment value. You configure where the data is stored by using the Source registry entry.

When you configure the Transaction ID Controller to use a file, the data is stored in the file and directory you specify in the registry.

When you configure the Transaction ID Controller to use the database, the data is stored in the IFW_TAM table, which is automatically created in the Pipeline Manager database by the Pipeline Manager installer.

Caution: If you configure the Transaction ID Controller to store IDs in the database, only one Pipeline Manager instance at a time can access the Pipeline Manager database. This can reduce transaction processing performance.

You should back up the transaction ID state file or table regularly. This state information is needed to ensure that your system continues to create unique, system-wide IDs when you stop and restart Pipeline Manager.

Configuring Sequence Checking

Sequence checking ensures that a CDR file is not processed more than once. You configure your Sequencers by using the **ifw.SequencerPool** registry entries, and you assign Sequencers to pipelines by using the pipeline **Output** registry entries. See "About the Sequencer" in *BRM Concepts*.

Sequence Numbers in the Header Record

The Header record in the EDR container includes two fields for sequence numbers:

- **SEQUENCE_NUMBER.** This is a unique reference that identifies each file. It indicates the file number of the specific file type, starting at 1 and incrementing by one for each new file of that type sent. Separate sequence numbering must be used for test and chargeable data. Having reached the maximum value (999999), the number restarts at 1.

Note: In the case of retransmission, this number is not incremented.

- **ORIGIN_SEQUENCE_NUMBER.** This is the original file sequence number as generated the first time. It is the same as **SEQUENCE_NUMBER**, but is never changed. It is used as a reference to the original file, if any processor has changed the file sequence number.

Deciding Whether to Use Sequencers

You should add Sequencers to your system when:

- You want to check for duplicate CDR files.
- Your CDR software does not automatically generate sequence numbers.
- Your pipelines split CDR files into multiple output files.

About Sequence Checking

When performing sequence checking, the Sequencer:

1. Receives the CDR file from the input module.
2. Checks for duplicates by comparing the sequence number in the stream's header with the sequence numbers in the state file or state table. See ["Sequencer Log Files and Log Tables"](#).
 - When the number is a duplicate, the Sequencer rejects the CDR file and rolls back the transaction.
 - When the number is not a duplicate, it passes the transaction directly to the Output Collection module. See "About Configuring the Output Section in the Registry" in *BRM Configuring Pipeline Rating and Discounting*.
3. Checks for gaps in sequence numbers by comparing the sequence number in the stream's header with the last sequence number in the state file or state table. If the sequence number is more than one digit greater than the previous number, a gap is identified. The Sequencer logs a message and stores the unused number in the state file or state table. See ["Sequencer State Files and State Tables"](#).

Note: By default, the Sequencer:

- Enables gaps in sequence numbers (caused by canceled or rolled back transactions). You can direct the Sequencer to reuse these number gaps by using the **Controller.ReuseGap** registry entry.
- Does not start the gap in sequence numbers from 0. For example, if the first sequence number is 3, the Sequencer does not start the gap for the skipped sequence numbers from 0 (that is, gap of 1, 2). You can direct the Sequencer to add a gap for the skipped sequence numbers starting from 0 by using the **Controller.UseGapAtStartup** registry entry.

See "Sequencer" in *BRM Configuring Pipeline Rating and Discounting*.

To configure the Sequencer to perform sequence checking, set the **SequencerType** registry entry to **Check**.

About Sequence Generation

When performing sequence generation, the Sequencer:

1. Receives the CDR file from the input module.
2. Assigns the next sequence number to the output file. To obtain this number, the Sequencer reads the last generated sequence number in the state file or state table and increments it by one.

This process continues for each CDR file until the maximum value is reached. For information, see ["About Maximum and Minimum Sequence Numbers"](#).

Note: If you configure the Sequencer to reuse gap numbers, it assigns unused gap numbers to the output file before assigning new sequence numbers. See "Sequencer" in *BRM Configuring Pipeline Rating and Discounting*

To configure the Sequencer to perform sequence generation, set the SequencerType registry entry to **Generation**.

About Maximum and Minimum Sequence Numbers

The Sequencer generates numbers by starting at the configured minimum value and then incrementing by one until it reaches the configured maximum value. After the Sequencer uses the maximum value, you must manually reset the sequence number to the minimum value.

For example, if the minimum value is 1 and the maximum value is 10,000, the Sequencer assigns 1 to the first output file, 2 to the second output file, 3 to the third output file, and so on. When the sequencer assigns 10,000 to the ten-thousandth output file, you must manually reset the sequence number to 1 by changing the following fields in the IFW_SEQCHECK table:

- Set the **seq_orignumber** field to 0.
- Set the **seq_gapnumbers** field to -1.

Important: To prevent the Sequencer from incorrectly rejecting files as duplicates after you manually reset the sequence number to the minimum value, remove all the rows from the IFW_SEQLOG_IN table.

To configure the maximum and minimum values, do one of the following:

- **State files.** Edit the **MaxSequenceNumber** and **MinSequenceNumber** entries in the state file. The default minimum value is 0; the default maximum value is 99999.
- **State tables.** Use Pricing Center to set these values as described for defining a sequence generation in *BRM Pricing Center Online Help*).

About Recycled EDRs

CDR input files sometimes contain nonvalid EDRs, which are rejected by the pipeline. When you recycle the input file through a pipeline to process any rejected EDRs, the file's original sequence number is no longer correct. The Sequencer automatically assigns new sequence numbers to recycled files to prevent them from being rejected as duplicates.

For more information about recycling, see "About Standard Recycling" in PI and "Recycling EDRs in Pipeline-Only Systems" in *BRM Configuring Pipeline Rating and Discounting*.

About Sequencer Files and Tables

Each Sequencer generates its own state and logging information, which can be stored in files or tables. You configure where state and logging information is stored by using the registry file. For information, see "Sequencer" in *BRM Configuring Pipeline Rating and Discounting*.

Important: When you store state and logging information in files, the Sequencer checks for duplicates by comparing the current sequence number against the last checked sequence number only. When you use tables, the Sequencer compares the number against all previously checked sequence numbers. For this reason, Oracle recommends using tables for production systems and using files only when testing your system in a development environment.

When you configure Sequencers to store logging information in files, all logging and state data is stored in the file and directory you specify in the registry file.

When you configure Sequencers to use tables, all logging and state data is stored in the database tables listed in [Table 7-11](#), which are automatically created by the Pipeline Manager installer:

Table 7-11 Sequencer Logging and State Data Database Tables

Table name	Description
IFW_PIPELINE	Stores information about pipelines.
IFW_SEQCHECK	Stores the state of the Sequencer.
IFW_SEQLOG_OUT	Stores sequence generation log information.
IFW_SEQLOG_IN	Stores sequence checking log information.

You use Pricing Center to provide input to IFW_SEQCHECK and to view log information stored in IFW_SEQLOG_OUT and IFW_SEQLOG_IN. See Pricing Center Help.

Sequencer State Files and State Tables

Sequencer state files and state tables store the following information:

- The last generated sequence number
- The last checked sequence number
- Maximum and minimum sequence numbers

You should back up state files and state tables periodically. This information is needed to ensure that your system does not process duplicate CDR files when you stop and restart Pipeline Manager.

Sequencer Log Files and Log Tables

Sequencer log files and log tables store an entry for each sequence number that is checked or generated.

Important: When the Sequencer reaches the maximum generated sequence number, delete all log entries. Otherwise, your log will contain duplicates. For more information, see "[About Maximum and Minimum Sequence Numbers](#)".

Tip: Log files and log tables grow indefinitely, so you should trim them periodically to reduce disk usage.

Checking and Generating Sequence Numbers

You can use Sequencers to configure pipelines to check for duplicate CDR input files and to check for gaps in sequence numbers. You can also configure pipelines to use Sequencers to generate sequence numbers. For information, see "[Configuring Sequence Checking](#)".

To enable sequence checking or sequence generation in a pipeline, perform the following tasks:

1. Configure your Sequencers by editing the **SequencerPool** section of the registry file. Ensure you specify the following:

- The Sequencer name.
- Whether Sequencer data is stored in a database table or files.
- How to connect to the database or the path and file name of the Sequencer files.
- Whether the Sequencer performs sequence checking or sequence generation. Each Sequencer performs only one of these functions.

For information, see "Sequencer" in *BRM Configuring Pipeline Rating and Discounting*.

2. For sequence generation, set minimum and maximum sequence numbers by doing one of the following:

- If you configured the Sequencer to store data in a *database*, use Pricing Center to set these values. See Pricing Center Help.
- If you configured the Sequencer to store data in *files*, set the **MaxSequenceNumber** and **MinSequenceNumber** entries in the Sequencer state file. For information, see "[About Maximum and Minimum Sequence Numbers](#)".

Note: The default minimum value is 0, and the default maximum value is 99999.

3. Assign Sequencers to pipeline output streams:

- To assign a sequence checker to an output stream, edit the Sequencer registry entry in the Pipeline Output Controller. Specify the name of the Sequencer assigned to the output stream:

```
Output
{
    ...
    Sequencer = SequenceCheckerName
```

```
...
}
```

For information, see "Output Controller" in *BRM Configuring Pipeline Rating and Discounting*.

- To assign a sequence generator to an output stream, edit the Sequencer registry entry in the output module. Specify the name of the Sequencer assigned to the output stream:

```
OutputStreamName
{
    ModuleName = OUT_GenericStream
    Module
    {
        Sequencer = SequenceGeneratorName
    }
}
```

For information, see "OUT_GenericStream" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring the NET_EM Module for Real-Time Processing

You can use Pipeline Manager for real-time discounting, real-time zoning, and real-time rerating. See "About the Pipeline Manager System Architecture" in *BRM Concepts*.

The NET_EM module provides a link between the Connection Manager (CM) and the pipelines. You configure the NET_EM module in the data pool.

To configure the NET_EM module, you configure connection information such as the port number and threads, and you configure the **OpcodeName** section for each type of real-time processing: discounting, rerating, and zoning.

In this example, you configure the real-time discounting by specifying the PCM_OP_RATE_DISCOUNT_EVENT opcode:

```
ifw
{
    ...
    DataPool
    {
        RealtimePipeline
        {
            ModuleName = NET_EM
            Module
            {
                ThreadPool
                {
                    Port = 14579
                    UnixSockFile = /tmp/rerating_em_port
                    Threads = 2
                }
                DiscountOpcode
                {
                    OpcodeName = PCM_OP_RATE_DISCOUNT_EVENT
                    NumberOfRTPipelines = 2
                    PipelineName = DiscountPipeline
                }
            }
        }
    }
}
```

```
}

```

Each NET_EM module can perform one type of processing; for example, discounting, rerating, or zoning. You must configure a separate instance of Pipeline Manager for each NET_EM module.

You can configure multiple instances of the same type of NET_EM processing, for example, multiple rerating Pipeline Manager instances. You can then configure the CM to point to all the NET_EM modules. When multiple rerating pipeline instances are configured, the NET_EM module routes rerate requests to whichever of these pipeline instances is available.

To configure the NET_EM module:

1. Configure the NET_EM module in the registry. See ["Configuring the NET_EM Module"](#).
2. Configure the CM to send data to the NET_EM module. See ["Configuring the CM to Send Real-Time Requests to the NET_EM Module"](#).

Configuring the NET_EM Module

The NET_EM module receives various types of requests from the CM and routes the requests to the appropriate pipeline. See "NET_EM" in *BRM Configuring Pipeline Rating and Discounting*.

Specifying the Type of NET_EM Opcode Processing

To specify the type of processing the NET_EM module is used for, use the **OpcodeName** entry.

For real-time discounting, use:

```
OpcodeName = PCM_OP_RATE_DISCOUNT_EVENT
```

For real-time zoning, use:

```
OpcodeName = PCM_OP_RATE_GET_ZONEMAP_INFO
```

For real-time rerating, use:

```
OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
```

Configuring the CM to Send Real-Time Requests to the NET_EM Module

To configure the CM to send rerate requests to the NET_EM module:

1. Open the CM configuration file (*BRM_home/sys/cm/pin.conf*).
2. For real-time rerating, ensure the following entry is *uncommented*:


```
- cm fm_module BRM_home/lib/fm_rerate.so fm_rerate_config - pin
```
3. Edit the discounting **em_group** entry:


```
- cm em_group em_type Opcode_name
```

where:

- *em_type* is the type of real-time processing; for example, discounting, zoning, or rerating. You can enter any string up to 15 characters. This entry must match the entry in the **em_pointer** entry.
- *Opcode_name* is the opcode used.

For discounting, use:

```
- cm em_group discounting PCM_OP_RATE_DISCOUNT_EVENT
```

For zoning, use:

```
- cm em_group zoning PCM_OP_RATE_GET_ZONEMAP_INFO
```

For rerating, use:

```
- cm em_group rating PCM_OP_RATE_PIPELINE_EVENT
```

4. Edit the discounting **em_pointer** entry to match your environment, for example:

```
- cm em_pointer discounting ip cm_host 11945
- cm em_pointer zoning ip cm_host 11945
- cm em_pointer rating ip cm_host 11945
```

Instructions for this entry are included in the file.

You can enter multiple **em_pointer** entries. If the first NET_EM module is unavailable, the CM connects to a different NET_EM module.

Note: To run multiple NET_EM instances, you must run multiple instances of Pipeline Manager. You use only one NET_EM module for each instance of Pipeline Manager.

5. Save the file.
6. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).

About Pipeline Manager Log Files

The log module is an optional pipeline component that generates and manages your system log files, which consist of the logs listed in [Table 7-12](#):

Table 7-12 Pipeline Manager Log Files

Log file	Description
Process log	Contains general system messages for the pipeline framework, such as startup, shutdown, version numbers of modules, and semaphore file messages. The module generates one process log for the entire pipeline framework.
Pipeline log	Contains messages for one pipeline, such as the opening and closing of batch files, the number of processed EDRs, and statistics. The module generates one pipeline log file per pipeline.
Stream log	Contains detailed messages for one output stream. The module generates one stream log file per input stream. It contains all single error messages for the stream and event; for example, <i>zone data not found</i> . Note: The number of stream log files grows indefinitely, so you should delete them periodically to save disk space.

You configure your system log files by editing the registry file. You create a set of log module registry entries for each type of log file you want your system to generate. For example, to configure your system to generate all three system log files, you create one set of entries for the process log, one set for the pipeline log, and one set for the stream log.

- You configure the process log in the **ProcessLog** registry section.
- You configure the pipeline log in the **PipelineLog** registry section for each pipeline.
- You configure the stream log in the **OutputLog** registry section for each pipeline.

For information, see "LOG" in *BRM Configuring Pipeline Rating and Discounting*.

In addition to the log files handled by the log module:

- All processed sequence numbers of the EDR streams are logged in the sequence log file. See "[Sequencer Log Files and Log Tables](#)".
- All processed transactions are logged in the transaction log file. See "[About Transaction Log Files](#)".

Pipeline Manager Log File Registry Entries

The registry entries listed in [Table 7–13](#) control Pipeline Manager log files.

Table 7–13 Pipeline Manager Log File Registry Entries

Entry	Module	Log file	Description
BalanceLockStatusLog	DAT_BalanceBatch	Process log	Specifies that when an event transaction is locked by an EDR transaction, it is logged to the Process log.
BinaryLogFileName	Transaction Manager	User specified	Specifies the path and file name of the binary log file, which is used to persist and restore open transactions.
InfranetPool	DAT_ConnectionPool		Specifies whether to log debug messages.
LogEvents	DAT_AccountBatch DAT_BalanceBatch DAT_Listener DAT_PriceModel DAT_Rateplan DAT_Recycle DAT_ResubmitBatch	Pipeline log	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling.
Logging	FCT_Opcode	Pipeline log	Logs each opcode called from the processing pipeline.
LogTestResults	FCT_Suspense		Determines whether the results of test recycling are logged.

Table 7–13 (Cont.) Pipeline Manager Log File Registry Entries

Entry	Module	Log file	Description
LogTransactions	DAT_BalanceBatch	Process log	Specifies if the balances affected during the CDR processing are logged.
LogZoneModelNotFoundEntries	FCT_USC_Map	Stream log	Specifies that all log entries in INF_NO_USC_MAPPING_ENTRY are logged into the Stream log.
RecycleLog	FCT_Recycle		Specifies the log file parameters.
WriteToLogEnabled	Transaction Manager	Pipeline log	Specifies whether the Transaction Manager writes status information to the pipeline log file.

See also ["Collecting Diagnostic Information by Using RDA"](#).

About Error Message Files

You use error message files to define the errors generated by your pipeline modules. All modules have their own error message file (**.msg**), which is installed by default in the *Pipeline_home/etc* directory.

The default error message files already define all of the module error codes, but you can add custom error codes or change the existing definitions by editing the files.

Error message file entries use the following format:

```
[messageName] | [messageText] | [messageNumber]
```

where:

- *messageName* specifies the module error code. For example, ERR_WRITE_FILE.
- *messageText* specifies the message text to write to the log file.
- *messageNumber* specifies the error number to write to the log file. The default is 0.

For example, the DAT_AccountBatch module uses the *Pipeline_home/etc/DAT_AccountBatch.msg* message file. This file includes the following entries:

```
ERR_LISTENER_NOT_FOUND | Listener '%s' not found.|30013
INF_STARTED_LOADING | Started loading account data.|30024
INF_ENTRIES_LOADED | %s %s loaded.|30025
INF_FINISHED_LOADING | Finished loading account data.|30026
```

Note: The LOG module ignores comments, which start with a pound symbol (#).

About Log File Contents

The LOG module logs the following information to the system log file in ITO format:

- Date
- Time
- Node
- Application name
- Message group
- Severity
- Error number
- Text

Note: All fields are separated by blanks.

For example:

```
03.10.2002 08:18:42 system ifw INTEGRATE NORMAL
00000 - No registry entry 'MultiThreaded(default is true)' found.
```

Troubleshooting Pipeline Modules

You can troubleshoot problems in the pipeline modules by writing the contents of the EDRs generated by various pipeline modules into a log file. The file shows how each module accessed the EDR and the changes each module made to the EDR. You can read the log file to check if the pipeline modules processed the EDRs as expected and correct any problems you find.

Use the **EdrTrace** entry in the pipeline registry file to write the contents of the EDR to a file. You can configure **EdrTrace** to write the EDR contents to a file for specific modules that you want to debug. The **EdrTrace** entry includes the parameters listed in [Table 7-14](#):

Table 7-14 *EdrTrace Log File Registry Entries*

Entry	Description
EDRTraceEnabled	Enables or disables EDR trace: <ul style="list-style-type: none"> ■ True enables EDR trace. ■ False disables EDR trace. The default is False .
EdrTrace	Specifies the EDR trace.
TraceLog	Specifies the following information about the EDR log file: <ul style="list-style-type: none"> ■ FilePath. The path to the log file. The default is /ifw/log/edrLog. ■ FileName. The name of the log file. The default is edrdump. ■ FilePrefix. The prefix to the log file name. The default is log_. ■ FileSuffix. The log file name extension. The default is .log.
TraceStartPoint	Specifies the pipeline module from which you want to start logging the EDR contents. This registry entry is mandatory. The default is Input.module .
TraceEndPoint	Specifies the pipeline module up to which you want to log the EDR contents. The default is Output.module . Important: If both the TraceStartPoint and TraceEndPoint registry entries are specified, the EDR log file contains changes from all the modules from TraceStartPoint to TraceEndPoint . If only TraceStartPoint is specified, the EDR log file contains changes from the module specified in that entry up to the Output module. To log EDR changes for only one module, TraceStartPoint and TraceEndPoint must specify the same module.

Writing EDR Contents to a Log File

To write the contents of the EDR to a log file and use it to debug pipeline modules, include the **EdrTrace** entry by using the following syntax:

```
...
Output
{
...
  EdrTraceEnabled = value
  EdrTrace
  {
    TraceLog
    {
      FilePath = file_path
      FileName = file_name
      FilePrefix = prefix
      FileSuffix = suffix
    }
    TraceStartPoint = Functions.Processing.FunctionPool.start_module_name
    TraceEndPoint = Functions.Processing.FunctionPool.end_module_name
  }
}
```

where:

- *start_module_name* is the user-defined name or label of the pipeline module from where the logging of the EDR contents starts.
- *end_module_name* is the user-defined name or label of the last pipeline module for the logging of the EDR contents.

Using a Semaphore to Write EDR Contents to a File for Debugging

You can change the EDR trace by sending a semaphore to the Output Controller module at run time without stopping the pipeline. You can perform the following changes to the **EdrTrace** entry through a semaphore:

- Enable or disable logging the EDR contents.
- Change **TraceStartPoint** and **TraceEndPoint** for logging the EDR contents.

To change the EDR content logging at run time, send a semaphore with the following syntax:

```
ifw.Pipelines.pipeline_name.Output.EdrTrace
{
  TraceStartPoint = new_start_value
  TraceEndPoint = new_end_value
}
```

Sample EDR Content Log File

The following sample output of **EdrTrace** shows EDR contents from Input to Output modules:

```
= = = = BEGIN TRANSACTION = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME : : test2.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.TRANSACTION_ID : 0.0 : 4 : setDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageType.Module : INTERNAL.TRANSACTION_ID
: 4 : : getDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloDiscountModule.Module :
INTERNAL.TRANSACTION_ID : 4 : : getDecimal
```

```

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloApplyBalanceModule.Module :
INTERNAL.TRANSACTION_ID : 4 : : getDecimal
al
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceOutputSplit.Module :
INTERNAL.TRANSACTION_ID : 4 : : getDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ObjectCacheTypeOutputSplit.Module :
INTERNAL.TRANSACTION_ID : 4 : : getDec
imal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Rejection.Module : INTERNAL.TRANSACTION_ID
: 4 : : getDecimal
ifw.Pipelines.ALL_RATE.Output : INTERNAL.TRANSACTION_ID : 4 : : getDecimal
= = = = B E G I N   C O N T A I N E R = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME : : test2.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.SEQ_CHECK : 0 : 1 : setLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : INTERNAL.STREAM_NAME
: test2.edr : : getString
ifw.Pipelines.ALL_RATE.Output : INTERNAL.STREAM_NAME : test2.edr : : getString
ifw.Pipelines.ALL_RATE.Output : INTERNAL.STREAM_NAME : test2.edr : : getString
ifw.Pipelines.ALL_RATE.Output : INTERNAL.SEQ_GENERATION : 0 : : getLong
ifw.Pipelines.ALL_RATE.Output : INTERNAL.OFFSET_GENERATION : 0 : : getLong
= = = = C O N T A I N E R   H E A D E R = = = =
ifw.Pipelines.ALL_RATE.Input : HEADER.TRANSFER_CUTOFF_TIMESTAMP : 20061204000445 : : getDate
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST : : : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST : : : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST : 00 : 00 : setString
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST : 00 : : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST : 49 : 49 : setString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST : 49 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : HEADER.QUERYABLE_
FIELDS_MAPPING : : : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : HEADER.CREATION_
PROCESS : PREPROCESS_PIPELINE : : get
String
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : HEADER.BATCH_ID : :
: getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Suspense.Module : HEADER.BATCH_ID : : :
getString
= = = = C O N T A I N E R   D E T A I L = = = =
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.DURATION : 300 : : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_END_TIMESTAMP : 20061115102400 : 20061115102400 :
setDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_END_TIMESTAMP : 20061115102400 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.NE_CHARGING_END_TIMESTAMP : 20061115102400 : 20061115102400 :
setDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.NE_CHARGING_END_TIMESTAMP : 20061115102400 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.RETAIL_CHARGED_AMOUNT_VALUE : 0.0 : : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE : 0.0 : : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 : : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_TYPE_OF_NUMBER : 0 : : getLong
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_MODIFICATION_INDICATOR : 00 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 : 0049100052 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_TYPE_OF_NUMBER : 0 : : getLong
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_MODIFICATION_INDICATOR : 00 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 : : getString

```

```

ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 : 0049100056 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.C_NUMBER : : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.RECORD_TYPE : 020 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_A_NUMBER_ZONE : 0049100052 : 0049100052 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_A_NUMBER_ZONE : 0049100052 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 : : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_B_NUMBER_ZONE : 0049100056 : 0049100056 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_B_NUMBER_ZONE : 0049100056 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : DETAIL.INTERN_
PROCESS_STATUS : 0 : : getLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : DETAIL.ASS_SUSPENSE_
EXT.PIPELINE_NAME.0 : : ALL_RATE
: setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : DETAIL.ASS_SUSPENSE_
EXT.SOURCE_FILENAME.0 : : test3.e
dr : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : DETAIL.ASS_SUSPENSE_
EXT.QUERYABLE_FIELDS.0 : : : set
String
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.BASIC_
SERVICE : TEL : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.INTERN_
USAGE_CLASS : : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.ASS_GSMW_
EXT.LOCATION_AREA_INDICATOR.0 : :
: getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.QOS_
REQUESTED : : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.QOS_USED :
: : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.RECORD_TYPE
: 020 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.INTERN_
SERVICE_CODE : TEL : TEL : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.INTERN_
SERVICE_CLASS : DEF : DEF : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.USAGE_CLASS
: NORM : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.USAGE_TYPE :
: : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.WHOLESALE_
IMPACT_CATEGORY : : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.TARIFF_CLASS
: : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.TARIFF_SUB_
CLASS : : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.RECORD_TYPE
: 020 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.CONNECT_TYPE
: 17 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.CONNECT_SUB_
TYPE : 01 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.INTERN_C_
NUMBER_ZONE : : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.USAGE_CLASS
: NORM : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.INTERN_
USAGE_CLASS : : NORM : setString

```

```

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module : DETAIL.DISCARDING
: 0 : : getLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module : DETAIL.RECORD_
TYPE : 020 : : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module : DETAIL.SOURCE_
NETWORK : : : getString
= = = = E N D   C O N T A I N E R = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME : : test3.edr : setString
= = = = E N D   T R A N S A C T I O N = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME : : test3.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.TRANSACTION_ID : 0.0 : 6 : setDecimal

```

Using Perl Scripts to Administer Pipeline Manager

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks.

Table 7–15 describes the files and scripts used for controlling pipelines:

Table 7–15 Pipeline Manager Administration Perl Scripts

Semaphore and Perl script file names	Description
dump_portal_act_data.reg dump_portal_act_data.pl	Outputs account data for all accounts currently in memory. By default, data is written to the cust.data file, located in the directory where you launch Pipeline Manager. Runs the DAT_Account module PrintData semaphore.
off_queue_buffer.reg off_queue_buffer.pl	Disables logging of the messages processed by the queue. Sets the DAT_Listener module LogEvents entry to False .
reload_portal_act_data.reg reload_portal_act_data.pl	Reloads accounts from the BRM database. Runs the DAT_Account module Reload semaphore.
reload_price.reg reload_price.pl	Reloads all the price models and rate plans. Runs the DAT_Price module Reload semaphore.
reload_zone.reg reload_zone.pl	Reloads all the zones and zone model data. Runs the DAT_Zone module Reload semaphore.
set_call_ass_limit.reg set_call_ass_limit.pl	Sets a new flush limit for call assembly (by default, 30). Runs the FCT_CallAssembling module FlushLimit semaphore.
set_dup_check_limit.reg set_dup_check_limit.pl	Sets a new limit for duplicate checking. If you do not specify any parameter, it sets the BufferLimit entry to three days before the current date, and it sets the StoreLimit entry to seven days before the BufferLimit date. This script creates and runs the set_dup_check_limit.reg semaphore. To modify the default BufferLimit and StoreLimit values, run the script with these two parameters: set_dup_check_limit.pl buffer_limit store_limit For example: set_dup_check_limit.pl 5 5 In this example, if today is November 28, then the buffer limit is set to November 23 and the store limit is set to November 18.

Controlling Batch Operations

This chapter explains how to configure the *Batch Controller* to launch *batch handlers* to check or process data for your Oracle Communications Billing and Revenue Management (BRM) system.

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 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.

Tip: 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:mypass@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**, using password **mypass**. 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.

Important: 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 can also check the Pipeline Manager registry to see if any temporary file output patterns conflict with this entry.

Caution: 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.

This example demonstrates the occurrence parameters:

```
batch.random.events          event1,event3
. . .
event1.name                  Random Discover file arrival
event1.handlers              handler1
event1.file.location         /apps/discover/stagingDir
event1.file.pattern          *.*.6011.*
. . .
event3.name                  Random Visa file arrival
event3.handlers              handler3
event3.file.location         /apps/visa/stagingDir
event3.file.pattern          *.dat
. . .
handler1.name                Discover UEL Handler
handler1.max.at.lowload.time 6
handler1.max.at.highload.time 3
handler1.start.string        /apps/discover -uel
. . .
handler3.name                Visa UEL Handler
handler3.max.at.lowload.time 8
handler3.max.at.highload.time 4
handler3.start.string        /apps/visa -uel
```

In this example, **event1** is triggered when any file name in the **/apps/discover/stagingDir** directory matches the pattern ***.*.6011.***. This match causes the Batch Controller to issue the command **/apps/discover -uel**, which runs **handler1**, the **Discover UEL Handler** program. Based on the parameters shown here, the Batch Controller starts no more than six concurrent instances of this program during the specified period of low expected load, or three during the high-load period.

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 execute 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.

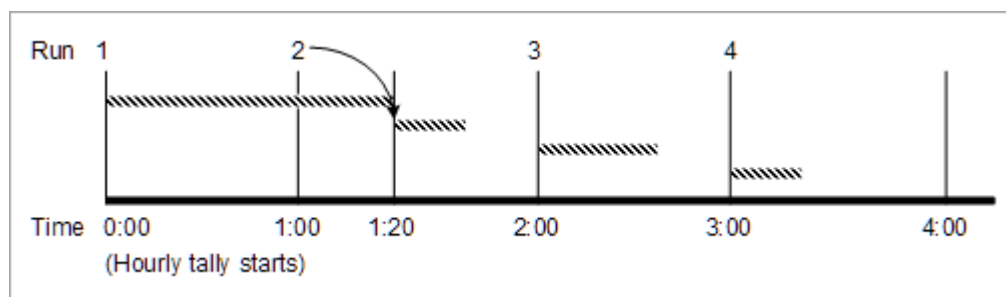
Caution: You must specify **batch.timed.events** or **batch.random.events** or both. Specifying neither causes the Batch Controller to shut down just after it starts because it has no tasks to perform.

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
```

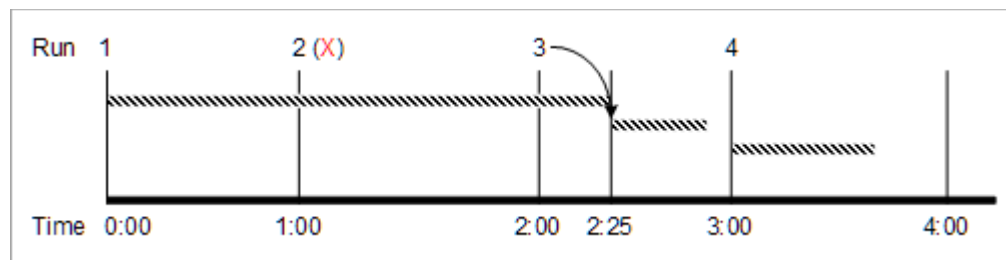
In this example, the occurrence specified as **event4** is named **Hourly tally**. It runs **handler4** for the first time at midnight (000000), and then runs it again every hour (3600 seconds) after that until it has run four times. If it cannot run at a scheduled time because previous executions are not finished, it runs again immediately as soon as possible. For example, consider the timeline in [Figure 8–1](#) for **event4**, above:

Figure 8–1 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 8–2](#):

Figure 8–2 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 *h:mm:ss*, 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.

Important: 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 can also check the Pipeline Manager registry to see if any temporary file output patterns conflict with this entry.

This example demonstrates the schedule parameters:

```
batch.timed.events      event5
. . .
event5.name             Sporadic tally
event5.handlers         handler5
event5.at               004500, 022500, 045500
event5.file.location    /apps/discover/stagingDir
event5.file.pattern     *.*.6011.*
. . .
```

handler5.name	Sporadic Handler
handler5.max.at.lowload.time	8
handler5.max.at.highload.time	4
handler5.start.string	/apps/sporadic-uel

In this example, the program specified as **event5** is named **Sporadic tally**. It runs only at 12:45 a.m., 2:25 a.m., and 4:55 a.m. If a program cannot run at a scheduled time because previous executions are not finished, it runs as soon as possible.

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.
- **sample_template.xml**: used by Universal Event (UE) Loader only.
- **SampleHandler.pl**: the actual code of the batch handler.
- **SampleHandler_config.values**: the batch handler's configuration file for application-specific parameters.
- **SampleReload.pl**: used by UE Loader only.

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 to Connect and Configure Components"](#).
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.

Important: 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.

Tip: 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 test system.

About Connection Pooling

This chapter describes Oracle Communications Billing and Revenue Management (BRM) connection pool functionality.

Overview

A connection pool is a set of connections maintained between an application, such as Content SDK, 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 is complete, 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. If a connection does not 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.

[Table 9-1](#) describes the configurable connection pool parameters in the **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, such as "Using Content SDK" in *BRM Content Manager*.

Infranet.properties File Connection Pool Parameters

Table 9–1 Configurable Connection Pool Properties

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

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 9–2](#):

Table 9–2 Connection Pool Error Handling

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

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

```
get toString()
```

Note: 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.

System Administration Utilities and Scripts

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) system administration utilities.

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: You cannot load separate */config/event_record_map* objects for each brand. All brands use the same object.

Important: 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.

Important: 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
```

Important: 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 to not 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 this 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

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 "Enabling Different Classes for Partitioning during Installation" in *BRM Installation Guide* and "Converting Nonpartitioned Classes to Partitioned Classes" in *BRM System Administrator's Guide*.

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"](#).

Important: 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.

For more information, see the following topics:

- [Partitioning Tables](#)
- [About Purging Data](#)

Location

BRM_home/bin

Syntax for Adding Partitions

```
partition_utils  -o add -t realtime|delayed -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

Adds real-time or delayed-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, 01032016.)

-u month|week|day

Specifies the time unit for the 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, to create one partition with a February 1 start date and the table already contains a P_R_02282016 partition, the P_R_02282016 partition is split into two partitions named P_R_02012016 and P_R_02282016.

-c storable_class

Specifies the class of objects to be stored in the partition. The default is */event*.

Note: To add a partition for a storable class other than an event, bill, invoice, item, or journal, partitioning must have been enabled for that storable class. See "Enabling Different Classes for Partitioning during Installation" in *BRM Installation Guide* and "Converting Nonpartitioned Classes to Partitioned Classes" in *BRM System Administrator's Guide*.

-w width

Specifies the number of units in a partition (for example, 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.

Caution: Before forcing partitions:

For real-time partitions, stop all BRM server processes.

For delayed-event partitions, stop all delayed-event loading by stopping Pipeline Manager and the Rated Event (RE) Loader utility (**pin_rel**).

Note: The **-f** parameter works differently when you remove partitions. In that case, it forces the removal of objects associated with open items.

-p

Writes 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"](#).

-b

Creates backdated partitions. For more information, see "Creating Partitions for Your Legacy Data" in *BRM Managing Customers*.

-h

Displays the syntax and parameters for this utility.

Syntax for Purging Objects without Removing Partitions

```
partition_utils -o purge [-s start_date] -e end_date -t [realtime|delayed]
[-p] [-h]
```

Only event, item, bill, invoice, journal, newsfeed, and sepa objects can be purged without removing their partitions.

To purge other types of objects, see ["Syntax for Removing Partitions"](#).

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.

-t realtime|delayed

Purges real-time objects, delayed-event objects, or both. The default is to purge both (include **-t** without **realtime** or **delayed**). If this parameter is omitted from the syntax, an error occurs.

-p

Writes 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"](#).

-h

Displays the syntax and parameters for this utility.

Syntax for Removing Partitions

```
partition_utils -o remove -s start_date -e end_date [-c storable_class]
               [-t realtime|delayed]
               [-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.

Note: To purge bill, event, invoice, item, journal, newsfeed, and sepa objects that meet the purging criteria, see ["Syntax for Purging Objects without Removing Partitions"](#).

To purge bill, event, invoice, item, journal, newsfeed, and sepa objects that do not meet the purging criteria, use the **-f** parameter, which removes the partitions that contain them.

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.

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"](#).

-e end_date

Specifies the end of the date range for the 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 the partitioned tables for the specified base storable class and its subclasses.

-t *realtimedelayed*

Removes real-time or delayed-event partitions. The default is to remove both real-time and delayed-event partitions.

-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.

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 time period of the current partition.

-p

Writes 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"](#).

-h

Displays the syntax and parameters for this utility.

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 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"](#).

-h

Displays the syntax and parameters for this utility.

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 be updated. The default is */event*.

-p

Writes 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"](#).

-h

Displays the syntax and parameters for this utility.

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 but cleans the status of it.

-h

Displays the syntax and parameters for this utility.

Syntax for Finding the Maximum POID for a Date

```
partition_utils -o maxpoid -s date -t realtime|delayed [-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

Gets the maximum POID in only real-time partitions or only delayed-event partitions.

-p

Writes 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"](#).

-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"](#).

Important: 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 Additional 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_clean_asos

Use this utility to delete closed **/active_session** objects from Oracle In-Memory Database (IMDB) Cache.

This utility compares an object's expiration time to the current time to determine if the object must be deleted and deletes the objects from one IMDB Cache at a time.

Note: The default **pin.conf** file for this utility includes the entry - **pin_mta multi_db**. This utility does not use this entry.

In a multischema environment, you must run this utility separately for each IMDB Cache node. You can create a script that calls the utility with connection parameters to connect to the desired node.

For more information about deleting expired objects from IMDB Cache, see "Purging Old Call Data from Memory" in *BRM Telco Integration*.

Important: To connect to the BRM database, this utility needs a **pin.conf** configuration file in the directory from which you run the utility.

Location

BRM_home/bin

Syntax

```
pin_clean_asos -object "object_type" [-expiration_time number_of_hours] [-help]
```

Parameters

-object "object_type"

Specifies the type of **/active_session** object to delete. For example, to delete **/active_session/telco/gsm** objects from IMDB Cache:

```
pin_clean_asos -object "/active_session/telco/gsm"
```

-expiration_time number_of_hours

Sets the expiration time (in hours) for **/active_session** objects. This utility compares the expiration time with an object's end time (PIN_FLD_END_T) and deletes objects that are older than the number of hours you specify. For example, if you specify **2** as the value and you run the utility at 10 a.m., the utility deletes objects that were closed on or before 8 a.m.

If not specified, the expiration time defaults to **0**. This results in the removal of all **/active_session** objects that have an end time less than the current time.

-help

Displays the syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

pin_clean_rsvns

Use this utility to free space in Oracle In-Memory Database (IMDB) Cache or the database by deleting objects that remain because of a session's abnormal termination, such as a missed stop accounting or cancel authorization request.

Note: This utility is installed with Resource Reservation Manager. See "About Resource Reservation Manager" in *BRM Configuring and Collecting Payments*.

This utility performs the following functions:

- Releases expired reservations

Note: When searching for reservation objects, this utility releases only expired reservations that are in reserved (active) status.

- Rates active sessions that are in a started or an updated state and updates balances in the database

Note: You can optionally rate active sessions in a created state.

- Deletes expired active-session objects

This utility compares an object's expiration time to the current time to determine if the object must be deleted and deletes objects from one IMDB Cache at a time.

Note: This utility requires a **pin.conf** file to provide entries for connecting to the CM and DM, login name and password, log file information, and performance tuning. The values for these entries are obtained from the information you provided during installation, or the entries are set with a default value.

The default **pin.conf** file for this utility includes the entry - **pin_mta_multi_db**. This utility does not use this entry.

In a multischema environment, you must run this utility separately for each IMDB Cache node. You can create a script to call the utility with connection parameters to connect to the desired node.

Location

BRM_home/bin

Syntax

```
pin_clean_rsvns [-help] [-object object_type] [-account]
                [-expiration_time number_of_hours] [-cause user_defined]
                [-bytes_uplink volume] [-bytes_downlink volume]
```

Parameters

-help

Displays the syntax and parameters for this utility.

-object *object_type*

Specifies the object storage location. In an IMDB Cache DM environment, *object_type* can only be 1.

-account

Calls STOP_ACCOUNTING to delete the active-session objects in both created and update states, and starts rating the session.

Use this parameter to rate active sessions if your network supports *only* the create state, and not start and update states, for sessions. For example, networks using the Message Based Interface (MBI) protocol send only authorization and reauthorization requests, so the active-session objects remain in a created state even during usage.

Note: When you run this utility without **- account**, it calls STOP_ACCOUNTING to delete the active-session objects in the update state.

-expiration_time *number_of_hours*

Sets back the expiration time for the objects, in hours.

The default is the current time when the utility runs. This utility deletes objects that are in an expired state up to the time you specify. For example, if you specify 2 as the value and you run the utility at 10 a.m., the utility deletes objects that expired by 8 a.m.

-cause *user_defined*

Specifies the reason for releasing reservations and terminating the session.

You can define any value for the cause, and the value is stored in the event object. You can define different rate plans depending on the reason for the session termination.

Note: You use the rate plan selector to specify the rate plan for calculating the charges for the event.

-bytes_uplink *volume*

When the */active_session* object's PIN_FLD_BYTES_UPLINK field is set to 0 or not specified, this parameter specifies to populate the field with the specified volume.

-bytes_downlink *volume*

When the */active_session* object's PIN_FLD_BYTES_DOWNLINK field is set to 0 or not specified, this parameter specifies to populate the field with the specified volume.

Results

The utility releases any expired reservations and deletes any expired session objects. If there are any session objects in a started or updated state, the utility rates the objects:

- For duration RUMs, it calculates the duration to rate by using the end time specified in the session object, the expiration time in the reservation object, the **pin_virtual_time**, or the current system time.

- For volume RUMs, it calculates the volume to rate by using the volume in the session object or the volume passed in at the command line.

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"](#).

Important: 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.

Important: 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.

For information on setting up and configuring the processes that this utility controls, see ["Configuring the pin_ctl Utility"](#).

For general information on creating configuration files, see ["Creating Configuration Files for BRM Utilities"](#).

Syntax

```
pin_ctl action component [-c file_name] [-collectdata] [-debug] [-i]
```

Parameters

action

Specifies the type of action to be executed. See ["action Parameter"](#).

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 ["component Parameter"](#).

-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

Enables the utility to ask whether to proceed. This is especially useful when running **stop**, **halt**, and **clear**.

action Parameter

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.

snmpget action

Gets an SNMP value.

To use this parameter, you must add **snmpget** actions by editing the **pin_ctl.conf** file. See ["Customizing snmpset and snmpget Actions"](#).

snmpset action

Sets an SNMP value:

- **addServerInstance.** Rebalances DM connection load when you restart a failed DM in the DM pool or when you add a new DM to the DM pool.
- **refreshConnections.** Rebalances DM connections when you restart a failed DM in the pool.

You can add **snmpset** actions by editing the **pin_ctl.conf** file. See ["Customizing snmpset and snmpget Actions"](#).

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 Components Included in "all""](#).

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 Components Included in "all""](#).

component Parameter

You can perform an action on any of the following components:

all

Applies action to the components specified in the **pin_ctl.conf** file. By default, the components are:

- 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 Components Included in "all"."](#)

answer

Paymentech answer simulator

batch_controller

Batch Controller

bre

Pipeline Manager

bre_tt

Pipeline Manager with IMDB Cache

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_ifw_sync

Account Synchronization DM (Oracle AQ)

dm_invoice

Invoice DM

dm_ldap

LDAP DM

dm_oracle

Oracle DM

dm_tt

Oracle In-Memory Database (IMDB) Cache DM

dm_vertex

Vertex tax calculation DM

eai_js

EAI Java Server

formatter

Invoice formatter

infmgr

System Manager

nmgr

Node Manager

rtp

Real-time pipeline

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 ["Using the pin_db_alert Utility to Monitor Key Performance Indicators"](#).

Important: 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_purge

Use this utility to delete old bills, items, journals, and expired account subbalances from Oracle In-Memory Database (IMDB) Cache.

This utility compares an object's expiration time to the current time to determine if the object must be deleted.

This utility deletes objects from one Oracle IMDB Cache or logical partition at a time. If the system has multiple logical partitions, you must run this utility for each logical partition. In a high-availability configuration, you must run this utility for each high-availability node. You can create a script to call this utility with relevant connection parameters to connect to the desired Oracle IMDB Cache nodes.

Important: Before running this utility, unset the ORACLE_HOME environment variable.

Note: This utility requires a **pin.conf** file in the directory from which you run the utility with entries for connecting to the CM and IMDB Cache DM, login name and password to connect to the data store, batch size for deleting, and log file information. The values for these entries are obtained from the information you provided during installation, or the entries are set with a default value. The **pin.conf** file for this utility is installed in the *BRM_home/apps/pin_subscription* directory.

Location

BRM_home/bin

Syntax

```
pin_purge [-l username/password@DatabaseAlias] -c { bill | item | subbalance }  
{-n number_of_days | -d date} [-help]
```

Parameters

-l username/password@DatabaseAlias

Specifies how to connect to the database. If you omit this option, the utility uses the information provided in the **pin.conf** file to establish the connection.

-c {bill | item | subbalance}

Specifies the object to be deleted.

- **-c bill** deletes bills and all related items and journals that do not have any pending or open items, due amounts set to zero, and bill status set to closed (bill in finalized)

Deleting bill objects does not also delete the associated bill items. The items must be deleted separately using the **-c item** parameter.

Note: The bill objects are deleted from Oracle IMDB Cache *only*.

- **-c item** deletes billable and special items (such as payments, adjustments, and disputes) in closed status and all journals related to these items.

Note: The item and journal objects are deleted from Oracle IMDB Cache *only*.

- **-c subbalance** deletes account subbalances.

Note: Account subbalances are deleted from Oracle IMDB Cache. The objects are eventually deleted from the BRM database when updates from Oracle IMDB Cache are propagated to the BRM database.

-n number_of_days

Specifies the number of days before which bills, items, or subbalances are deleted. For example, specify **90** to delete bills, items, or subbalances older than 90 days.

-d date

Specifies the date before which bills, items, or subbalances are deleted. Use the format *MM/DD/YYYY*. For example, specify **03/01/2009** to delete bills, items, or subbalances older than March 1, 2009.

-help

Displays the syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the **pin_purge.pinlog** 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.

pin_sub_balance_cleanup

Use this utility to purge expired account subbalances from the BRM database.

Caution: When you delete subbalances from the database, events that impacted those subbalances cannot be rerated. Ensure you no longer need the expired subbalances before deleting them.

For more information, see "[About Purging Account Subbalances](#)".

To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility.

Important: 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.

Location

BRM_home/bin

Syntax

```
pin_sub_balance_cleanup -n number_of_days | -d date
```

Parameters

-n number_of_days

Specifies the number of days before which subbalances are deleted. For example, specify **60** to delete expired subbalances older than 60 days.

-d date

The date before which subbalances are deleted. Use the format *MM/DD/YYYY*. For example, specify **06/30/2003** to delete expired subbalances older than June 30, 2003.

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_tt_schema_gen

Use this utility to connect to the BRM database and generate SQL scripts to create and initialize the BRM cache groups in Oracle IMDB Cache. These SQL scripts contain all the required cache group definitions. You can run these SQL scripts against the appropriate IMDB Cache nodes to load the cache groups with the required schema and data. See ["Generating the BRM Cache Group Schema"](#).

Requirements

This utility requires the following versions of the Perl modules:

- DBI version 1.605
- DBD-Oracle version 1.16
- Bit-Vector version 7.1

Note: This utility has been certified for Perl 5.8.0 and Oracle 11g.

Before running this utility, configure the database connection parameters in one of the following files in *BRM_home/bin*.

- **pin_tt_schema_gen.values**, which generates scripts for the default cache groups
- A custom configuration values file, which generates scripts for custom cache groups

See ["Configuring the pin_tt_schema_gen.values File"](#) for more information.

Important: 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.

Note: When you run this utility, the following warning messages are logged in the **pin_schema_gen.log** file. These warnings are reported for storable classes that do not have associated tables. You can ignore these warning messages.

'/reservation/active' mentioned in array local_tables_class_def does not have any table.

'/active_session/telco/gprs/master' mentioned in array local_tables_class_def does not have any table.

'/active_session/telco/gprs/subsession' mentioned in array local_tables_class_def does not have any table.

Location

BRM_home/bin

Syntax

```
pin_tt_schema_gen [-t] [-l] [-d] [-o] [-a] [-h] [-f configuration_values_files ]
```

Parameters

-t

Generates the **tt_schema.sql** script file, which you can run against the appropriate IMDB Cache node to create the cache groups and transient table schema.

-l

Generates the **tt_load.sql** script file, which you can run against the appropriate IMDB Cache node to load data from the BRM database.

-d

Generates the **tt_drop.sql** script file, which you can run against the appropriate IMDB Cache node to drop the cache groups.

-o

Updates the BRM database with unique indexes and not-null constraints.

-a

Runs the **-t**, **-l**, **-d**, and **-o** parameters.

-h

Displays the syntax and parameters for this utility.

-f *configuration_values_files*

Specifies the name of the configuration values files to be used by the utility. The default is **pin_tt_schema_gen.values**. You can provide another configuration values file.

Results

If the utility does not notify you that it was successful, look in the **pin_tt_schema_gen.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.

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 create|pre_export|post_export|verify_types|create_
types[-execute]
                  -readtasks create|pre_export|post_export|verify_types|create_
types[-execute]
                  -showtasks [minID maxID]
                  -help
```

Parameters

-gentasks create [-execute]

Generates tasks and stores them in the database.

Executes tasks after saving to the database.

-readtasks create [-execute]

Reads previously stored tasks from the database.

Executes tasks after reading from the database.

-gentasks create

Creates virtual columns and supporting columns in the BRM database.

Use this with **-showtasks** to display the tasks that will be executed for creating the virtual columns before executing them. The following example shows how to create the tasks for creating virtual columns, display them, and then execute 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.

-readtasks pre_export [-execute]

Reads and then executes the previously stored pre_export tasks in the database.

-readtasks post_export [-execute]

Reads and then executes the previously stored post_export tasks in the database.

-readtasks verify_types [-execute]

Reads and then executes the previously stored verify_types tasks in the database.

-readtasks create_types [-execute]

Reads and then executes the previously stored create_types tasks in the database.

-help

Displays the syntax and parameters for this utility.

Results

The utility notifies you when it runs successfully. Otherwise, look in the **vcoll.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.

Important: 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 "Archiving Audit Data" in *BRM Developer's Guide*.

For information on audit trails and shadow objects, see "About Tracking Changes to Object Fields" in *BRM Developer's Guide*.

Location

BRM_home/sys/archive/oracle

Syntax

```
purge_audit_tables.pl report -t objects -d date -l login/pswd@connection|
create -t objects -l login/pswd@connection|
archivedirect -t objects -d date -c commit_size -l login/pswd@connection|
archiveindirect -t objects -d date -c commit_size -l login/pswd@connection|
renametohist -t objects -l login/pswd@connection|
updfromhist -t objects -d date -c commit_size -l login/pswd@connection|
help
```

The following purging actions are supported:

- [Syntax for Generating Audit Table Reports](#)
- [Syntax for Creating History Tables](#)
- [archivedirect Syntax](#)
- [archiveindirect Syntax](#)
- [renametohist Syntax](#)
- [updfromhist Syntax](#)

Syntax for Generating 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 login/pswd@connection
```

Parameters for Generating Audit Table Reports

-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 login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

Syntax for Creating History Tables

Creates empty history tables for the specified objects and their child objects.

```
purge_audit_tables.pl create -t objects -l login/pswd@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 10-1](#).

Table 10-1 Audit and History Tables Format

/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

Important: Do not specify the child objects for a table; they are handled automatically by the script.

-l login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

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 login/pswd@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 simultaneously.

-l login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

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.

Important: 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 login/pswd@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 simultaneously.

-l login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

renametohist Syntax

Renames the specified audit tables to their corresponding history tables and recreates 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 login/pswd@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 login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

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 login/pswd@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 simultaneously.

-l login/pswd@connection

Specifies your standard Pipeline Manager user name and database password.

help

Displays the syntax for this script.

Results

The utility notifies you only if it encounters errors.

SNMP Utilities

This chapter describes the syntax and parameters for the AGENT++ SNMP utilities that are installed with Oracle Communications Billing and Revenue Management (BRM).

Note: The `snmpTrap` and `snmpInform` utilities are not supported.

Important: Because the AGENT++ SNMP implementation uses dynamic OIDs, you cannot use symbolic SNMP names in SNMP commands. For example, instead of using this command:

```
snmpWalk <hostname> 10.196.129.31  
portal.components.mtf.connectionConfigurations.dmoTable.dmoEntry  
-P20761 -S
```

You must use this:

```
snmpWalk <hostname> 10.196.129.31 1.3.6.1.4.1.3512.1.5.2.2.1 -P20761 -S
```

For more information about SNMP commands and options, see the SNMP documentation provided by your OS vendor.

snmpBulk

Retrieves a subtree of OIDs. Uses the SNMPv2 GETBULK request to retrieve information from an SNMPv2C agent. If the node is an SNMPv1-only agent, this utility automatically turns the GETBULK request into an SNMPv1-supported GETNEXT request.

Syntax

```
snmpBulk IP_address | DNS_name [OID1 [OID2...]] [option1 [option2...]]
```

Parameters

IP_Address* | *DNS_Name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All objects in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-vN**
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-Pport**
Remote port to use.
- **-Ccommunity_name**
Specifies the community. The default is **public**.
- **-rN**
Specifies the number of retries to be used in the requests. The default is **1**.
- **-tN**
Specifies the timeout in hundredths of a second between retries. The default is **100**.
- **-nN**
Specifies nonrepeaters (number of object instances that should be retrieved no more than once from the beginning of the request. The default is **0**.
- **-mN**
The maximum number of times that other variables beyond those specified by the nonrepeaters field should be retrieved). The default is **1**.

snmpDiscover

Broadcasts a network discovery request to find out if the SNMP master agent is running.

Syntax

```
snmpDiscover Broadcast_IP_address [option1 [option2...]]
```

Parameters

Broadcast_IP_address

IP address of the network.

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.

snmpGet

Uses the SNMP GET request to query for information on a network entity.

Syntax

```
snmpGet IP_address | DNS_name [OID] [option1 [option2...]]
```

Parameters

IP_Address* | *DNS_Name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All objects in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.
- **-t*N***
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpNext

Gets data about the next object in the MIB. This utility sends an SNMP GETNEXT request to the master agent and waits for a response back before it proceeds.

Syntax

```
snmpNext IP_address | DNS_name [OID] [option1 [option2...]]
```

Parameters

IP_address* | *DNS_name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All objects in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.
- **-t*N***
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpNextAsync

Gets data about the next object in the MIB. This utility sends an SNMP GETNEXT request to the master agent and receives the response from a callback function. The main process does not have to wait for the response before it proceeds.

Syntax

```
snmpNextAsync IP_address | DNS_name [OID] [option1 [option2...]]
```

Parameters

IP_address* | *DNS_name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All variables in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.
- **-t*N***
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpPasswd

Sets a password.

Syntax

```
snmpPasswd IP_address | DNS_name user new_password [option1 [option2...]]
```

Parameters

IP_address* | *DNS_name

The location of the SNMP process.

user

The user name.

new_password

The new password.

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2** or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.
- **-t*N***
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpSet

Sets a value in an object.

Syntax

```
snmpSet IP_address | DNS_name [OID] [option1 [option2...]]
```

Parameters

IP_address* | *DNS_name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All objects in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-vN**
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-Pport**
Remote port to use.
- **-Ccommunity_name**
Specifies the community. The default is **public**.
- **-Gcommunity_name**
Specifies the GET community. The default is the SET community value.
- **-rN**
Specifies the number of retries to be used in the requests. The default is **1**.
- **-tN**
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpWalk

Retrieves a subtree of OIDs.

Syntax

```
snmpWalk IP_address | DNS_name [OID] [option1 [option2...]]
```

Parameters

IP_address* | *DNS_name

The location of the SNMP process.

OID

Specifies which part of the MIB is searched. All objects in the subtree below the given OID are queried.

The default is **sysDescr** (description of the host on which the SNMP agent is running).

options

Can be any of the following, separated by a space:

- **-vN**
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-Pport**
Remote port to use.
- **-S**
Walk only within the subtree.
- **-Ccommunity_name**
Specifies the community. The default is **public**.
- **-rN**
Specifies the number of retries to be used in the requests. The default is **1**.
- **-tN**
Specifies the timeout in hundredths of a second between retries. The default is **100**.

snmpWalkThreads

Retrieves a subtree of OIDs. The start OID is 1.

This utility gets all OID values from one or multiple SNMP master agents (one master agent per thread when processing the SNMP WALK request). **snmpWalk** gets the values on a subtree of OIDs from one SNMP master agent.

Syntax

```
snmpWalkThreads host/port [host/port ...] [option1 [option2...]]
```

Parameters

host/port

Host name and port of the system you are getting information about.

options

Can be any of the following, separated by a space:

- **-v*N***
Use SNMP version **1**, **2**, or **3**. The default is **1**.
- **-P*port***
Remote port to use.
- **-C*community_name***
Specifies the community. The default is **public**.
- **-r*N***
Specifies the number of retries to be used in the requests. The default is **1**.
- **-t*N***
Specifies the timeout in hundredths of a second between retries. The default is **100**.

Part II

Administering a High-Availability System

Part II describes how to configure an Oracle Communications Business and Revenue Management (BRM) high-availability system. It contains the following chapters:

- [Understanding a High-Availability System](#)
- [Configuring a High-Availability System](#)

Understanding a High-Availability System

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) high-availability architecture. In addition, it explains how the BRM components in a high-availability system handle failover.

For information about setting up a high-availability system, see "[Configuring a High-Availability System](#)".

For information about the standard BRM system architecture, see "BRM System Architecture" in *BRM Concepts*.

About High-Availability BRM Systems

A high-availability system is designed to continue functioning when one or more of its components fail. To do this, it contains backup components to which it automatically switches (fails over) when an active component stops working. The failover should appear seamless to users and should not interrupt service.

In a BRM system, high availability is required primarily for real-time processing of prepaid and postpaid services and for operations performed by customer service representatives (CSRs). Batch processes, such as invoicing and billing, typically do not require high availability because they do not involve real-time interaction with users. You can handle failure in batch processing by restarting the component or by accepting slower performance rather than by switching to a backup component. Therefore, this chapter covers high-availability systems for real-time processing only.

About the Architecture of a High-Availability BRM System

A high-availability BRM system contains one or more backup components for each of the following components:

- Connection Manager (CM)
- In-Memory Database Cache Data Manager (IMDB Cache DM)
- Oracle IMDB Cache (the data store)
- Oracle Clusterware (includes a built-in backup mechanism)
- Oracle Real Application Clusters (Oracle RAC) instance

The primary and backup components should not run on the same physical host or use the same power supply.

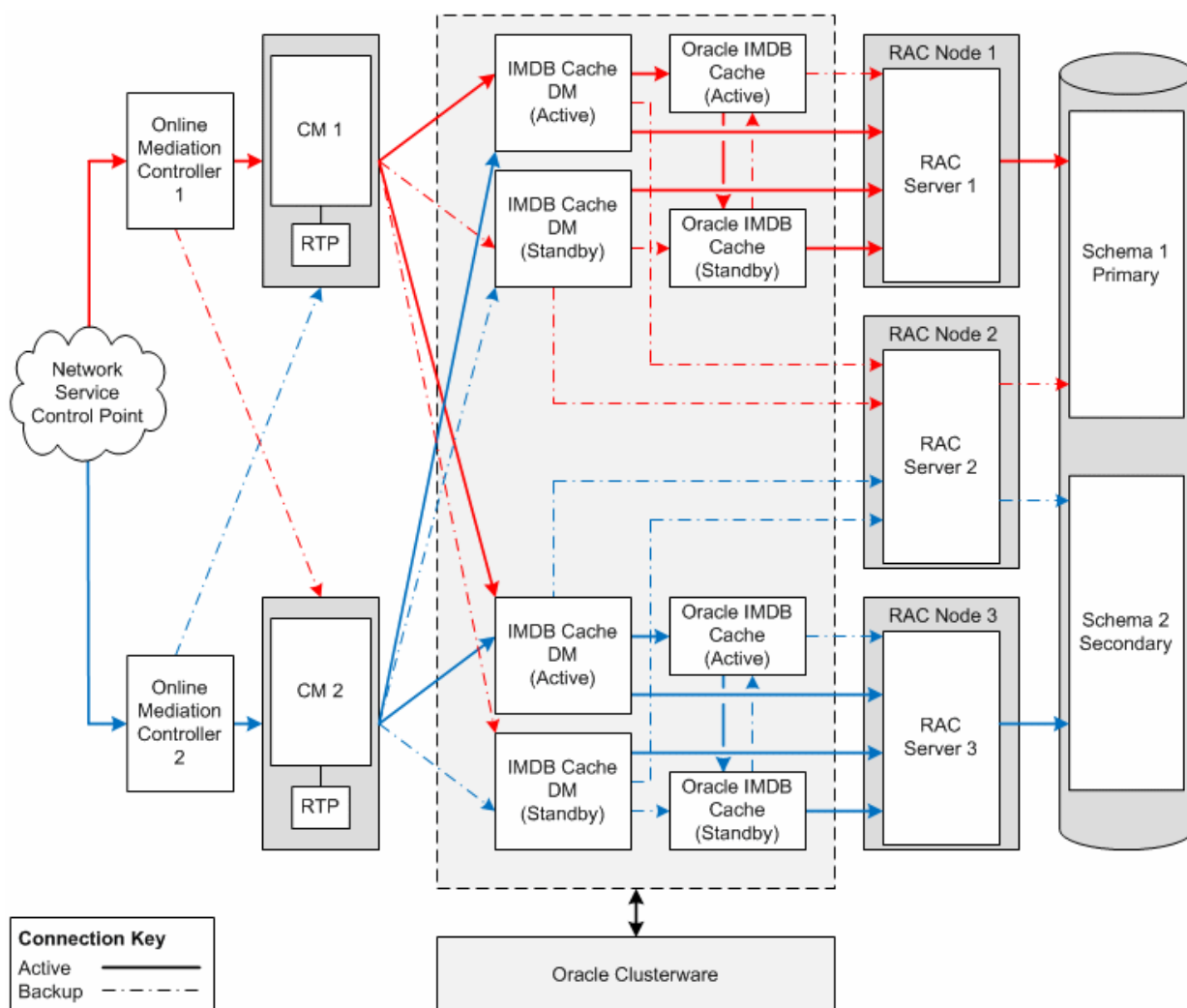
Each instance—primary and backup—of a component should be connected to all instances of the component's server-side peer (the component that it calls). For

example, each CM should be connected to each IMDB Cache DM. This ensures that if one instance of a component fails, another one is available to process data.

Note: In addition to backups, the level of availability of the components in a BRM system depends on the sizing that you use (that is, the number of computers in the system). The more computers that you spread your components across, the fewer components that are affected if one of the computers fails.

Figure 12–1 shows the architecture of the basic set of components required to create a high-availability BRM system. Dashed lines represent backup connections.

Figure 12–1 Basic High-Availability BRM System



Each component in a high-availability system starts the failover process in the following situations:

- A network connection to the underlying host server is lost.

- A response to a request is not received within the timeout period specified for the component that sent the request. The delay can be due to hardware, software, or network problems.

The following sections explain how each component in a high-availability BRM system handles failover:

- [About Connection Managers in a High-Availability System](#)
- [About IMDB Cache DMs and Data Stores in a High-Availability System](#)
- [About the BRM Database in a High-Availability System](#)

For information about setting up components in a high-availability system, see ["Configuring a High-Availability System"](#).

About Connection Managers in a High-Availability System

In a high-availability system, the CMs detect failures in the pipelines and in the IMDB Cache DMs to which they connect. This section describes how CMs handle such failures:

- [How CMs Handle Real-Time Pipeline Failure](#)
- [How CMs Handle IMDB Cache DM Failure](#)

For information about configuring CMs for a high-availability system, see ["Configuring Connection Managers for High Availability"](#).

How CMs Handle Real-Time Pipeline Failure

When a CM receives a request for rating that requires discounts to be applied or zoning to be considered by Pipeline Manager, the CM forwards the request to the instance of Pipeline Manager running real-time discounting and zoning pipelines. If the Pipeline Manager instance fails and does not send a response to the CM, the request times out, and the CM sends an error message to the calling application.

Note: If a request has no discounting or zoning requirements, the CM sends the request to the IMDB Cache DM.

How CMs Handle IMDB Cache DM Failure

In a high-availability system, the failure of an IMDB Cache DM is detected by a CM. CMs handle the following types of IMDB Cache DM failure:

- [Initial Connection Failure](#)
- [Session Failure](#)

For an overview of the IMDB Cache DM, see *BRM Concepts*.

For information about IMDB Cache DMs in a high-availability system, see ["About IMDB Cache DMs and Data Stores in a High-Availability System"](#).

Initial Connection Failure

In a high-availability system, each CM connects to an active and a standby IMDB Cache DM. If the active DM does not respond when the CM tries to establish a connection to it, the CM performs the following actions:

1. Checks the **dm_pointer** entries in the CM **pin.conf** file for the host name and port number of the standby IMDB Cache DM and connects to the standby DM.
2. Logs a failover message in the CM log file.
3. Initiates the IMDB Cache DM failover process by sending a login request to the standby DM with the PCM_OPFLG_RETRY flag.

The standby IMDB Cache DM accepts the login request and starts the failover process only if its associated Oracle IMDB Cache (data store) is active.

Otherwise, it rejects the login request with the PIN_ERR_NOT_ACTIVE error and forces the CM to retry the formerly active DM. (The PIN_ERR_NOT_ACTIVE error is recorded as PIN_ERR_NOT_PRIMARY in the CM log file.)

4. If the standby IMDB Cache DM rejects the CM's login request, the CM tries to connect to the formerly active IMDB Cache DM at intervals specified in the **pcm_bad_connection_retry_delay_time_in_secs** entry in the CM **pin.conf** and IMDB Cache DM **pin.conf** files.

Session Failure

If a CM loses its connection to the active IMDB Cache DM during a session or if a request from the CM to the active DM times out, the CM performs the following actions:

1. Closes the connection to the active IMDB Cache DM.
2. Logs a failure message in the CM log file.
3. Checks the **dm_pointer** entries in its **pin.conf** file for the name and port number of the standby IMDB Cache DM.
4. Initiates the IMDB Cache DM failover process by sending a login request to the standby DM with the PCM_OPFLG_RETRY flag.

- If the failed entry opcode is not in an explicit transaction, the CM retries the opcode with the PCM_OPFLG_RETRY flag.

Explicit transaction means that Oracle Communications Service Broker (OCSB) Online Mediation Controller (OCMC) sent a transaction open request to the CM before retrying the failed entry opcode.

- If the failed entry opcode is in an explicit transaction, the CM returns the PIN_ERRCLASS_SYSTEM_RETRYABLE error to OCMC.

OCMC ends the transaction and then retries the same transaction.

The standby DM accepts the request and starts the failover process only if the Oracle IMDB Cache (data store) that it connects to is active.

Otherwise, it rejects the request with the PIN_ERR_NOT_ACTIVE error and forces the CM to resend its request to the formerly active DM. (The PIN_ERR_NOT_ACTIVE error is recorded as PIN_ERR_NOT_PRIMARY in the CM log file.)

See ["How IMDB Cache DMs Handle Data Store Failure"](#).

5. If the standby DM rejects the request, the CM tries to reconnect to the formerly active DM and resume the interrupted session.

About IMDB Cache DMs and Data Stores in a High-Availability System

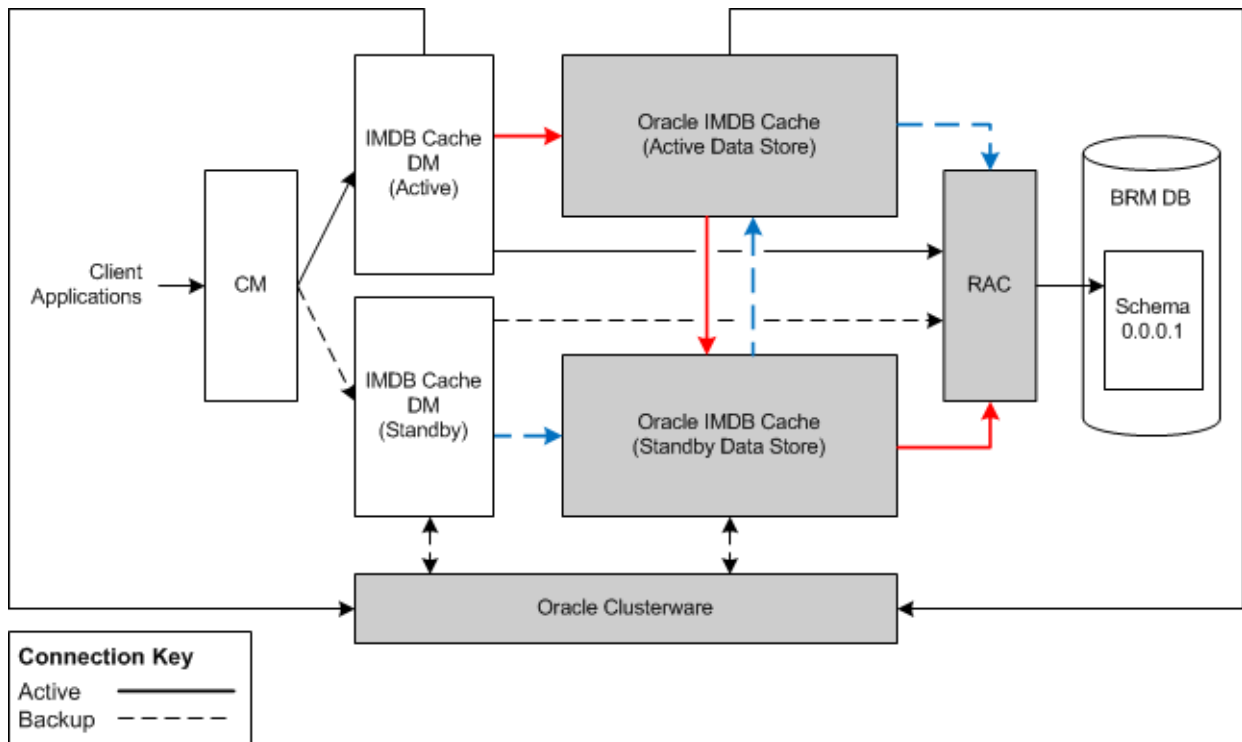
The basic BRM high-availability architecture has one pair of *active* and *standby* IMDB Cache DM instances and one pair of *active* and *standby* Oracle IMDB Cache instances

for each BRM database schema. Larger high-availability systems can have several IMDB Cache DMs and Oracle IMDB Cache pairs for each schema.

Note: An Oracle IMDB Cache instance is also called a *data store*.

Figure 12–2 shows a typical relationship between the IMDB Cache DMs and data stores in a high-availability system. The dashed line represents a backup connection.

Figure 12–2 Typical Relationship between IMDB Cache DMs and Data Stores in a High-Availability System



As the preceding figure illustrates, the active IMDB Cache DM is connected to the active data store and the standby IMDB Cache DM is connected to the standby data store. Both IMDB Cache DM instances are also directly connected to the Oracle RAC-enabled BRM database. The replication agent associated with the active data store propagates the updates to the standby data store. Updates in the standby data store are propagated to the BRM database.

When an IMDB Cache DM starts, it connects to its associated data store and checks the state of that data store. If the data store is active, the IMDB Cache DM sets its own processing state to active. If the data store is on standby, the IMDB Cache DM sets its processing state to standby.

The active IMDB Cache DM processes all requests, and only the data in the active data store is updated. The standby data store receives the updates from the active data store and propagates them to the BRM database.

The standby IMDB Cache DM processes requests only when the state of its data store changes to active.

In a high-availability system, the data store ensures data availability and integrity.

This section covers the following topics:

- [How IMDB Cache DMs Fail Over](#)
- [How IMDB Cache DMs Handle Data Store Failure](#)
- [How IMDB Cache DMs Handle Oracle RAC Failure](#)

For information on IMDB Cache DMs and data stores, see ["Using Oracle IMDB Cache Manager"](#).

For information on configuring IMDB Cache DMs for a high-availability system, see ["Configuring IMDB Cache Manager for High Availability"](#).

How IMDB Cache DMs Fail Over

An IMDB Cache DM failure can occur in the following situations:

- The node on which the active IMDB Cache DM and its data store reside fails. See ["About Active Node Failure"](#).
- The IMDB Cache DM associated with the active data store fails. See ["About Active IMDB Cache DM Failure"](#).

About Active Node Failure

In a high-availability system, an IMDB Cache DM instance and its associated data store reside on the same physical server (*node*).

When an active node fails, the failover process is as follows:

1. Oracle Clusterware detects the active data store failure and does the following:
 - Changes the state of the standby data store to active.
 - Changes the state of the formerly active data store to standby.
2. The CM sends requests to the standby IMDB Cache DM instance.
3. The standby IMDB Cache DM checks the state of its associated data store.
 - If the data store is active, the standby DM changes its own processing state to active and processes the request.
 - If the data store is on standby or if the standby DM's connection to the data store fails, the standby DM rejects the CM request with the `PIN_ERR_NOT_ACTIVE` error. (The error is recorded as `PIN_ERR_NOT_PRIMARY` in the CM log file.)

About Active IMDB Cache DM Failure

IMDB Cache DM failover is managed by Oracle Clusterware. When Oracle Clusterware detects an IMDB Cache DM failure, the failover process is as follows:

1. Oracle Clusterware tries to restart the formerly active DM.
2. While Oracle Clusterware is trying to restart the DM, the CM continues trying to connect to the DM at intervals for the duration of the specified retry time period (see ["pcm_bad_connection_retry_delay_time_in_secs"](#)).

If the CM cannot connect within the specified time period, it redirects requests to the standby IMDB Cache DM.

See ["How CMs Handle IMDB Cache DM Failure"](#).

3. If Oracle Clusterware can restart the formerly active DM, the DM checks the state of its associated data store.

- If the data store is active, the DM sets its processing state to active and starts processing CM requests.
 - If the data store is on standby, the IMDB Cache DM sets its processing state to standby.
4. If Oracle Clusterware cannot restart the formerly active DM, you must manually activate the standby DM. See ["Manually Activating a Standby IMDB Cache DM"](#).

Manually Activating a Standby IMDB Cache DM

When an active IMDB Cache DM fails, its associated data store is not notified of the failure, so the data store's status remains active. This prevents the standby data store from becoming active.

Because its associated data store is still on standby, the standby DM rejects all CM requests with the `PIN_ERR_NOT_ACTIVE` error to indicate that it is in standby mode and not accepting requests. (The `PIN_ERR_NOT_ACTIVE` error is recorded as `PIN_ERR_NOT_PRIMARY` in the CM log file.)

Therefore, if an internal IMDB Cache DM error prevents Oracle Clusterware from restarting a DM, you must manually change the standby data store's state to active. This enables the standby DM to switch its state to active and process the requests redirected to it by the CM.

Important: All CM requests will fail until either the active or standby IMDB Cache DM establishes a connection with an active data store.

How IMDB Cache DMs Handle Data Store Failure

When the active IMDB Cache DM receives a request from the CM, it passes the request to its associated data store only if that data store's state is active.

The data store is considered to have failed in the following situations:

- When the initial IMDB Cache DM connection to the data store fails. See ["Initial Connection Failure"](#).
- When the connection to the data store is lost during a transaction or a request has timed out. See ["Transaction Failure"](#).

Initial Connection Failure

When the active IMDB Cache DM's attempt to connect to its data store fails, the following actions occur:

1. The DM logs a connection failure message in the IMDB Cache DM log file.
2. The DM retries the connection to the data store.
3. If the retry connection fails, the DM rejects the CM request and logs a `PIN_ERR_STORAGE` error.
4. Oracle Clusterware detects the active data store failure and does the following:
 - Changes the state of the standby data store to active.
 - Changes the state of the formerly active data store to standby.
5. The CM sends the request to the standby IMDB Cache DM.
6. The standby DM checks the state of its associated data store.

- If the data store is active, the standby DM sets its processing state to active and processes the request.
- If the data store is on standby or if the connection to the data store fails, the standby DM rejects the CM request with the `PIN_ERR_NOT_ACTIVE` error. (The error is recorded as `PIN_ERR_NOT_PRIMARY` in the CM log file.)
- If the standby DM's connection to its data store fails, the CM tries to reconnect to the active IMDB Cache DM.

Transaction Failure

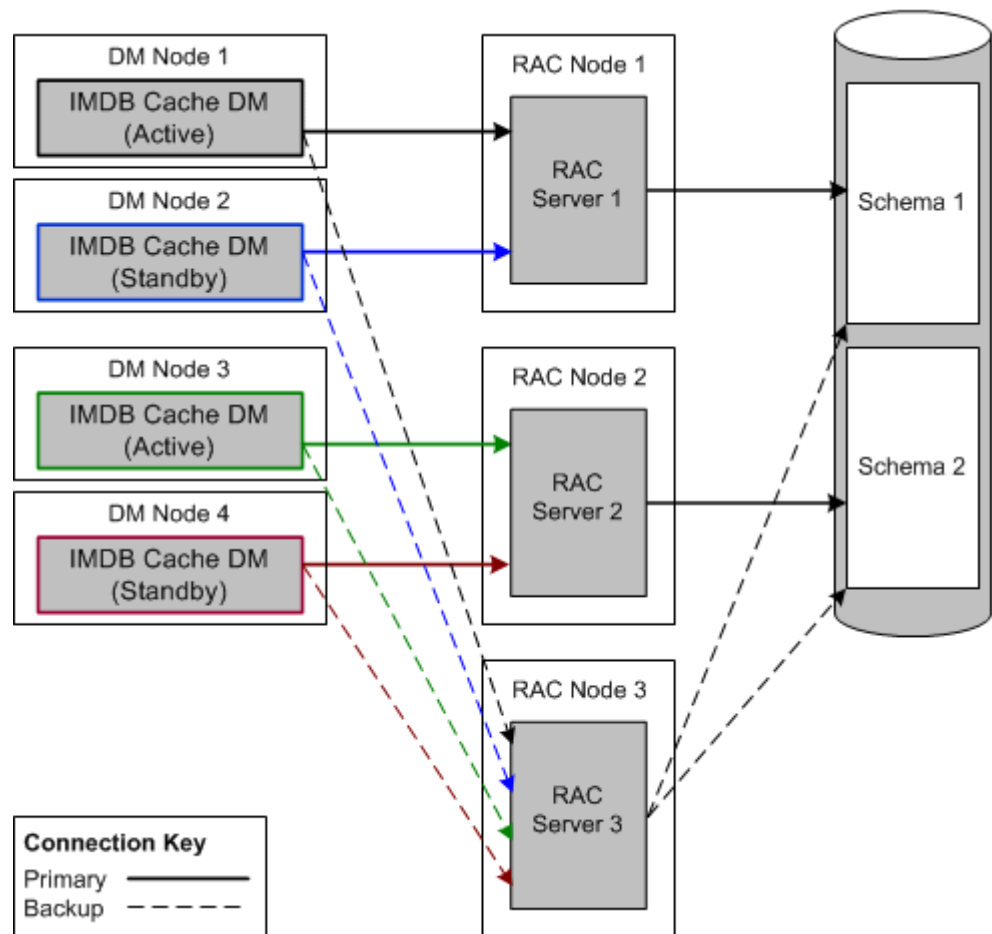
If the active IMDB Cache DM loses its connection with its data store during a transaction or if a request from the active DM to its data store times out, the following actions occur:

1. The DM logs a connection failure message in the IMDB Cache DM log file.
2. The DM retries the connection to its data store.
3. If the connection fails, the DM sets its processing state to standby and rejects the CM request with the `PIN_ERR_NOT_ACTIVE` error.
4. The CM sends the request to the original standby IMDB Cache DM.
5. Oracle Clusterware detects the active data store failure and does the following:
 - Changes the state of the original standby data store to active.
 - Changes the state of the formerly active data store to standby.
6. The original standby DM checks the state of its associated data store.
 - If the data store is active, the original standby DM sets its processing state to active and processes the request.
 - If the data store is on standby or if the connection to the data store fails, the original standby DM rejects the CM request with the `PIN_ERR_NOT_ACTIVE` error. (The error is recorded as `PIN_ERR_NOT_PRIMARY` in the CM log file.)

How IMDB Cache DMs Handle Oracle RAC Failure

In a high-availability system, each IMDB Cache DM maintains connections to a primary Oracle RAC node and to a backup Oracle RAC node through connect descriptors in the `tnsnames.ora` file referenced by the DM. The database service associated with the connect descriptor specifies which Oracle RAC node is primary and which is the backup for that DM. See ["Setting Up Oracle RAC for Failover in a High-Availability System"](#).

In [Figure 12-3](#), solid lines represent primary connections and dashed lines represent backup connections.

Figure 12-3 IMDB Cache DM Connections to Oracle RAC Nodes

In normal processing, each IMDB Cache DM sends requests to its primary Oracle RAC instance. If that Oracle RAC instance becomes unavailable, the IMDB Cache DM connects to its backup Oracle RAC instance. This section describes how IMDB Cache DMs handle Oracle RAC failure.

In a high-availability system, IMDB Cache DMs handle the following types of Oracle RAC failure:

- [Initial Connection Failure](#)
- [Session Failure](#)

Initial Connection Failure

Connection failures can be due to hardware, software, network, or listener problems or to a lack of response from the Oracle RAC instance. If an IMDB Cache DM's initial attempt to connect to its primary Oracle RAC instance fails, the IMDB Cache DM connects to its backup Oracle RAC instance.

Session Failure

When an Oracle RAC instance fails while processing a request, the IMDB Cache DM detects the failure and connects to its backup Oracle RAC instance. Any transaction that is active when the failure occurs is rolled back. Information about the failover is logged in the IMDB Cache DM **pinlog** file.

If an IMDB Cache DM loses the connection to its primary Oracle RAC instance in the middle of a session, it performs the following actions:

1. Tries to reestablish the connection to its primary Oracle RAC instance.
2. Performs one of the following actions:
 - If the reconnection attempt is successful, continues processing the request.
 - If the reconnection attempt is unsuccessful, clears any incomplete connection to the primary Oracle RAC instance and tries to connect to its backup Oracle RAC instance.

In the following situations, the IMDB Cache DM returns a `PIN_ERRCLASS_SYSTEM_RETRYABLE` error and `PIN_ERR_STORAGE` error code to the CM so the client can retry the transaction:

— The failover occurs in the middle of a transaction.

— The failover occurs outside a transaction, and the IMDB Cache DM *cannot* finish the operation.

If the failover happens outside a transaction and the IMDB Cache DM *can* finish the operation, the failover is transparent to the CM.

About the BRM Database in a High-Availability System

This section describes the database components of a high-availability system:

- [About Oracle Real Application Clusters and Oracle Clusterware](#)
- [About Multischema High-Availability Systems](#)

For an overview of multischema systems, see the discussion about the multischema architecture in *BRM Concepts*.

For information about configuring the BRM database in a high-availability system, see ["Configuring the BRM Database for High Availability"](#).

About Oracle Real Application Clusters and Oracle Clusterware

For a high-availability system, you must use Oracle RAC, which consists of multiple Oracle RAC instances. Each Oracle RAC instance has the following characteristics:

- Runs on its own cluster node and server
- Is typically associated with only one schema
- Concurrently processes data for a single database with all the other Oracle RAC instances in the cluster

Oracle RAC requires a highly available, high-speed storage system. A storage area network (SAN) running on clustered hardware is recommended. The cluster nodes are connected through a high-performance grid.

Oracle Clusterware is used to manage Oracle RAC servers. It also facilitates state management of Oracle IMDB Cache instances (data stores) and manages the failover of IMDB Cache DM instances by restarting the IMDB Cache DM process when it detects a failure.

For information about installing and configuring Oracle RAC instances and Oracle Clusterware, see the Oracle RAC and Oracle Clusterware documentation.

About Multischema High-Availability Systems

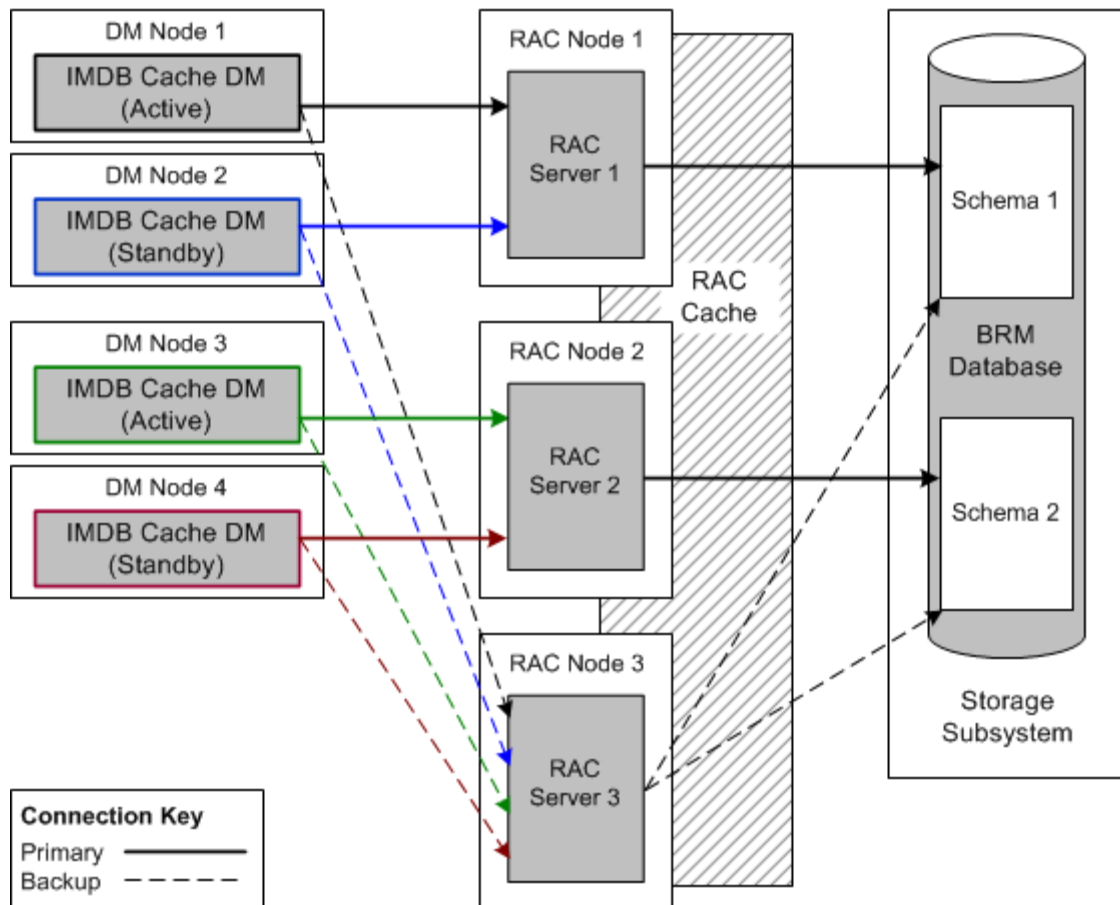
Multischema high-availability systems are built on an Oracle RAC system with one primary Oracle RAC instance for every schema in your system plus at least one backup Oracle RAC instance. The backup Oracle RAC instance can take over for any primary Oracle RAC instance that fails.

Each IMDB Cache DM is connected to a primary Oracle RAC instance and to the backup Oracle RAC instance. Because a primary Oracle RAC instance never handles the load of more than one server, all primary Oracle RAC servers can be sized to run at 80% capacity during peak processing times. This reduces your system's overall spare idle capacity.

Note: You can increase the number of backup Oracle RAC instances to meet business requirements.

Figure 12-4 shows the configuration for a multischema high-availability system with two schemas. Solid lines represent primary connections and dashed lines represent backup connections.

Figure 12-4 Multischema High-Availability System Configuration with Two Database Schemas



Configuring a High-Availability System

This chapter provides guidelines for configuring an Oracle Communications Billing and Revenue Management (BRM) high-availability system for real-time processing of prepaid and postpaid services.

For an overview of a high-availability BRM system, see ["Understanding a High-Availability System"](#).

About Setting Up a High-Availability System

To create a high-availability system, you must install and configure *at least two* instances of each component and then connect each instance to all instances of the component's server-side peer.

To create a high-availability BRM system, follow these guidelines:

- Install and configure the following components:
 - A single-schema or multiscHEMA BRM database.
 - One Oracle Real Application Clusters (Oracle RAC) instance for each database schema, and at least one additional Oracle RAC instance to use as a backup.
 - Oracle Clusterware.
 - One or more pairs of active and standby Oracle In-Memory Database Cache (Oracle IMDB Cache) instances for each database schema. (Oracle IMDB Cache instances are also called *data stores*.)
 - One or more pairs of active and standby IMDB Cache Data Managers (DMs) for each database schema.
 - At least two Connection Managers (CMs). Use as many as you estimate will support your workload, and then add one more.
 - One real-time pipeline for each CM.

Important: Duplicate components should reside on different physical hosts and use different power sources. They should not be located in the same rack. If you use blade servers, install duplicate components in different blade chassis. (This also applies to active and standby Oracle IMDB Cache and IMDB Cache DM instances.)

For more information about configuring components in a high-availability system, see the following sections:

- * [Configuring the BRM Database for High Availability](#)
 - * [Configuring IMDB Cache Manager for High Availability](#)
 - * [Configuring Connection Managers for High Availability](#)
- Connect each instance of a component to all instances of the component's server-side peer. For example, configure each CM to connect to all the IMDB Cache DMs.
 - When setting timeout periods for processing a request, ensure that each client-side (or calling) component has a longer timeout period than the server-side component that it calls.

The *timeout period* (also called *latency*) is the time taken by a component to respond to a request from the moment the request is sent to the component.

The timeout period should be long enough to accommodate slow responses due to overload, in which case there is an occasional response rather than no response.

[Table 13–1](#) lists the timeout settings that affect component failover in a high-availability system. The components are listed in the order that they process requests received from the network.

Table 13–1 Timeout Settings for High Availability Systems

Component	Timeout Setting	Description	Suggested Value
Connection Manager	pcm_timeout_in_msecs	Specifies the amount of time that the CM waits for a response from an IMDB Cache DM before failing over to the next IMDB Cache DM in the dm_pointer list. This entry is known as the CM's long (failover) timeout. See " Connecting CMs to IMDB Cache DMs ".	100 seconds
Connection Manager	pcm_bad_connection_retry_delay_time_in_secs	Specifies the interval at which the CM tries again to connect to the active IMDB Cache DM after it fails to connect. The timeout should be long enough for the CM to reestablish a connection with the IMDB Cache DM. See " Connecting CMs to IMDB Cache DMs ".	100 seconds
Oracle IMDB Cache	LockWait	Specifies the number of seconds that Oracle IMDB Cache (the data store) waits to acquire a lock for the BRM database before returning a timeout error to the IMDB Cache DM. See " Creating and Configuring Active Data Stores ".	30 seconds
Oracle RAC database	FAST_START_MTTR_TARGET	Specifies the time limit for Oracle RAC instance recovery. See " Minimizing Recovery Time of Oracle RAC Instances ".	30 seconds

For a diagram of the data processing flow in a high-availability system, see "[About the Architecture of a High-Availability BRM System](#)".

Important: Timeout periods must be large enough to accommodate slow responses because of overload.

Configuring the BRM Database for High Availability

The BRM database in high-availability systems consists of the following components:

- Oracle RAC. See "[About Oracle Real Application Clusters and Oracle Clusterware](#)".
- A single-schema or multischema BRM database. See "[About Multischema High-Availability Systems](#)".

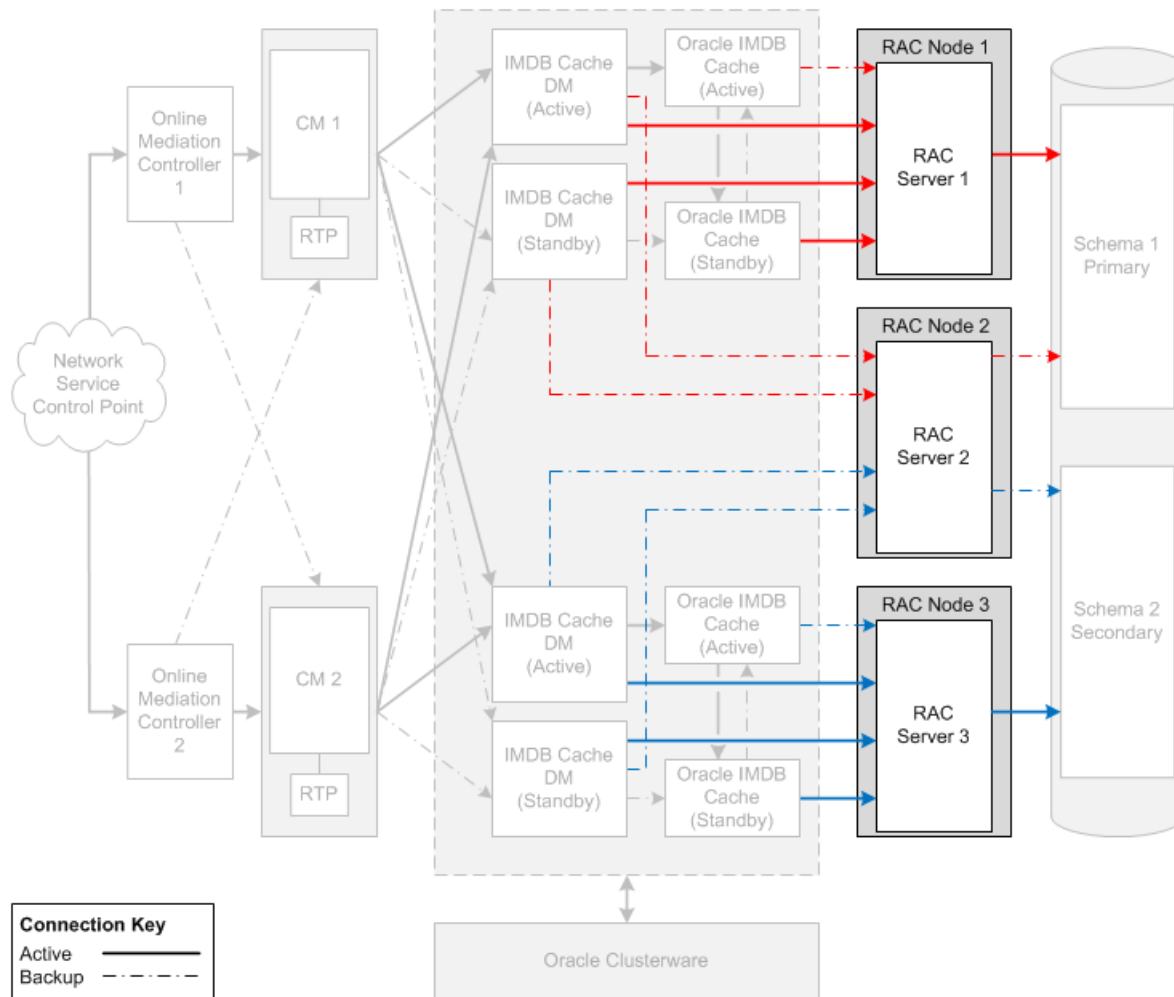
The following sections explain how to configure Oracle RAC for failover in a high-availability system:

- [Setting Up Oracle RAC for Failover in a High-Availability System](#)
- [Minimizing Recovery Time of Oracle RAC Instances](#)

Note: Using a standby database and database recovery are out of the scope of this chapter.

Setting Up Oracle RAC for Failover in a High-Availability System

[Figure 13–1](#) shows the configuration of Oracle RAC in a basic high-availability system. Dashed lines represent backup connections.

Figure 13–1 Oracle RAC Configuration for a Basic High-Availability System

To configure Oracle RAC for failover in a high-availability system:

1. Set up an Oracle RAC instance for each database schema in your system and then add at least one more Oracle RAC instance to function as the backup. For example, if you have three schemas, set up at least four Oracle RAC instances.

For information about how to set up an Oracle RAC system, see the Oracle RAC documentation.

Note: The number of backup Oracle RAC instances depends on your high-availability requirements and on the number of primary Oracle RAC instances. Typically, one backup Oracle RAC instance should be configured for every four or five primary Oracle RAC instances. (This recommendation assumes that after a failed Oracle RAC instance is fixed, you switch the database service that it originally supported back to it.)

The Oracle RAC instances should reside on different physical hosts and use different power sources.

2. Configure Oracle database services. See ["Configuring Oracle Database Services"](#).

3. Add entries for the Oracle database services to all **tnsnames.ora** files that will be referenced by the IMDB Cache DMs in your system. See ["Defining Connections to the Oracle Database Services"](#).
4. Configure your IMDB Cache DMs to connect to the Oracle RAC instances. See ["Connecting IMDB Cache DMs to Oracle RAC Instances for High Availability"](#).

Configuring Oracle Database Services

You use Oracle database services to connect IMDB Cache DMs to Oracle RAC instances. To create a high-availability system, you must map each database service to one primary Oracle RAC instance and to one backup Oracle RAC instance.

Note: In Oracle RAC systems, the primary Oracle RAC instance is called the *preferred instance*, and the backup Oracle RAC instance is called the *available instance*.

For example, if your system has four database schemas and five Oracle RAC instances, configure the database services as shown in [Table 13-2](#):

Table 13-2 Example Database Service Configuration

Database Service	Primary Oracle RAC Instance	Backup Oracle RAC Instance
Service1	Oracle RAC instance 1	Oracle RAC instance 5
Service2	Oracle RAC instance 2	Oracle RAC instance 5
Service3	Oracle RAC instance 3	Oracle RAC instance 5
Service4	Oracle RAC instance 4	Oracle RAC instance 5

To create the services in the preceding table, log on to any Oracle RAC node as the Oracle database administrator, and run the following commands:

```

srvctl add service -d racDatabaseName -s service1 -r racInstanceName1 -a racInstanceName5 -P Basic
srvctl add service -d racDatabaseName -s service2 -r racInstanceName2 -a racInstanceName5 -P Basic
srvctl add service -d racDatabaseName -s service3 -r racInstanceName3 -a racInstanceName5 -P Basic
srvctl add service -d racDatabaseName -s service4 -r racInstanceName4 -a racInstanceName5 -P Basic

```

For information about the **srvctl** command, see the Oracle RAC documentation.

You must also configure each service to be notified if the backup Oracle RAC instance becomes unreachable or network connections fail. To do this, enable Fast Application Notification (FAN) for each database service. For information about FAN, see *Automatic Workload Management with Oracle Real Application Clusters* on the Oracle Technology Network (<http://www.oracle.com/technetwork/index.html>)

Optionally, you can enable Transparent Application Failover (TAF) in Basic failover mode for the database services. TAF is not required by IMDB Cache DMs, but it benefits other BRM applications that connect directly to the database, such as Rated Event Loader. In Basic mode, applications connect to a backup Oracle RAC node only *after* their connection to the primary Oracle RAC node fails. This approach has low overhead, but end users might experience a delay while the new connection is created. For more information about TAF, see the Oracle database documentation.

Defining Connections to the Oracle Database Services

Perform the following procedure in the appropriate **tnsnames.ora** files.

To define connections to the Oracle database services:

1. Open the **tnsnames.ora** file in a text editor.

By default, that file is in the *Oracle_home/network/admin/* directory.

2. For each database service, add the following connect descriptor:

```
connectionString =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = primaryRacInstanceHostName) (PORT =
oraHostPortNo))
    (ADDRESS = (PROTOCOL = TCP) (HOST = backupRacInstanceHostName) (PORT =
oraHostPortNo))
    (LOAD_BALANCE = OFF)
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = serviceName)
      (FAILOVER_MODE =
        (TYPE = SELECT)
        (METHOD = BASIC)
        (RETRIES = 180)
        (DELAY = 5)
      )
    )
  )
```

where:

- *connectionString* is the connection name. The **sm_database** entry in the IMDB Cache DM **pin.conf** file must match this entry.
- **ADDRESS** defines a single listener protocol address. Add an entry for the primary Oracle RAC instance's listener and the backup Oracle RAC instance's listener.
- *primaryRacInstanceHostName* is the name of the computer on which the service's primary Oracle RAC instance resides.
- *backupRacInstanceHostName* is the name of the computer on which the service's backup Oracle RAC instance resides.
- *oraHostPortNo* is the port number for the Oracle database on the host computer. Typically, this number is 1521.
- **LOAD_BALANCE** specifies whether to distribute connection requests across the listeners specified in the **ADDRESS** entries. For high-availability BRM systems, set this to **OFF**.
- *serviceName* specifies the name of the database service. The **sm_svcname** entry in the IMDB Cache DM **pin.conf** file must match this entry.
- For **FAILOVER MODE**,
 - TYPE = SELECT** specifies that work in progress is not lost when failover occurs from the primary to the backup Oracle RAC node.
 - METHOD = BASIC** specifies that applications connect to a backup Oracle RAC node only *after* their connection to the primary Oracle RAC node fails. Backup connections are not preestablished.

3. Save and close the file.

Connecting IMDB Cache DMs to Oracle RAC Instances for High Availability

To connect each IMDB Cache DM to its primary and backup Oracle RAC instances in a high-availability BRM system:

1. Open the IMDB Cache DM configuration file in a text editor:

BRM_home/sys/dm_tt/pin.conf

2. Set the **sm_database** entry to the appropriate connect descriptor:

- **dm sm_database** *connectionString*

connectionString must match the appropriate *connectionString* entry in the **tnsnames.ora** file referenced by the IMDB Cache DM.

3. Add the following entry:

- **dm sm_svcname** *serviceName*

serviceName must match the **SERVICE_NAME** entry in the connect descriptor specified in the preceding step.

4. Save and close the file.
5. Stop and restart the IMDB Cache DM. See ["Starting and Stopping the BRM System"](#).
6. Repeat this procedure for each IMDB Cache DM in your high-availability system.

Minimizing Recovery Time of Oracle RAC Instances

To minimize the recovery time of Oracle RAC instances, use the **FAST_START_MTTR_TARGET** initialization parameter to reduce the size of the redo log file. When setting this parameter, you must balance system performance against failure recovery time. Use the following values:

- **0**—Disables the parameter. In this case, blocks containing modified data not yet written to disk (dirty blocks) are flushed mainly during checkpoints triggered by redo log switches, so instance recovery time depends primarily on the amount of redo log data to apply at the time of failure. If the redo logs are huge and the current log is almost full, recovery might take several hours.
- **1 through 3600**—Specifies the time limit in seconds for database instance recovery. To meet this target, the Oracle database adjusts the frequency of checkpoint creation. It might need to proactively flush dirty blocks from the database cache to disks. This requires additional input/output operations, which can degrade performance.

For example:

```
alter system set FAST_START_MTTR_TARGET=30;
```

The preceding command sets database instance recovery time to 30 seconds. This value is a recommended starting point, but you should test it in your environment to find the optimal setting.

Important: The timeout period of all BRM components in a high-availability system should be greater than **FAST_START_MTTR_TARGET**.

When `FAST_START_MTTR_TARGET` is set to a short time period, such as 30 seconds, you can further reduce service downtime by lowering the database cluster heartbeat interval. (By default, Oracle RAC waits 30 seconds before resetting a node after the loss of its heartbeat.) A very short heartbeat interval, however, might result in unnecessary node resets due to network blips.

Note: Failure of one Oracle RAC node interrupts service in all Oracle RAC nodes because Oracle RAC must remaster internal services and restore the database by using the current state of the redo log file.

For more information about redo log files, see "Assigning Storage for Redo Log Files" in *BRM Installation Guide*.

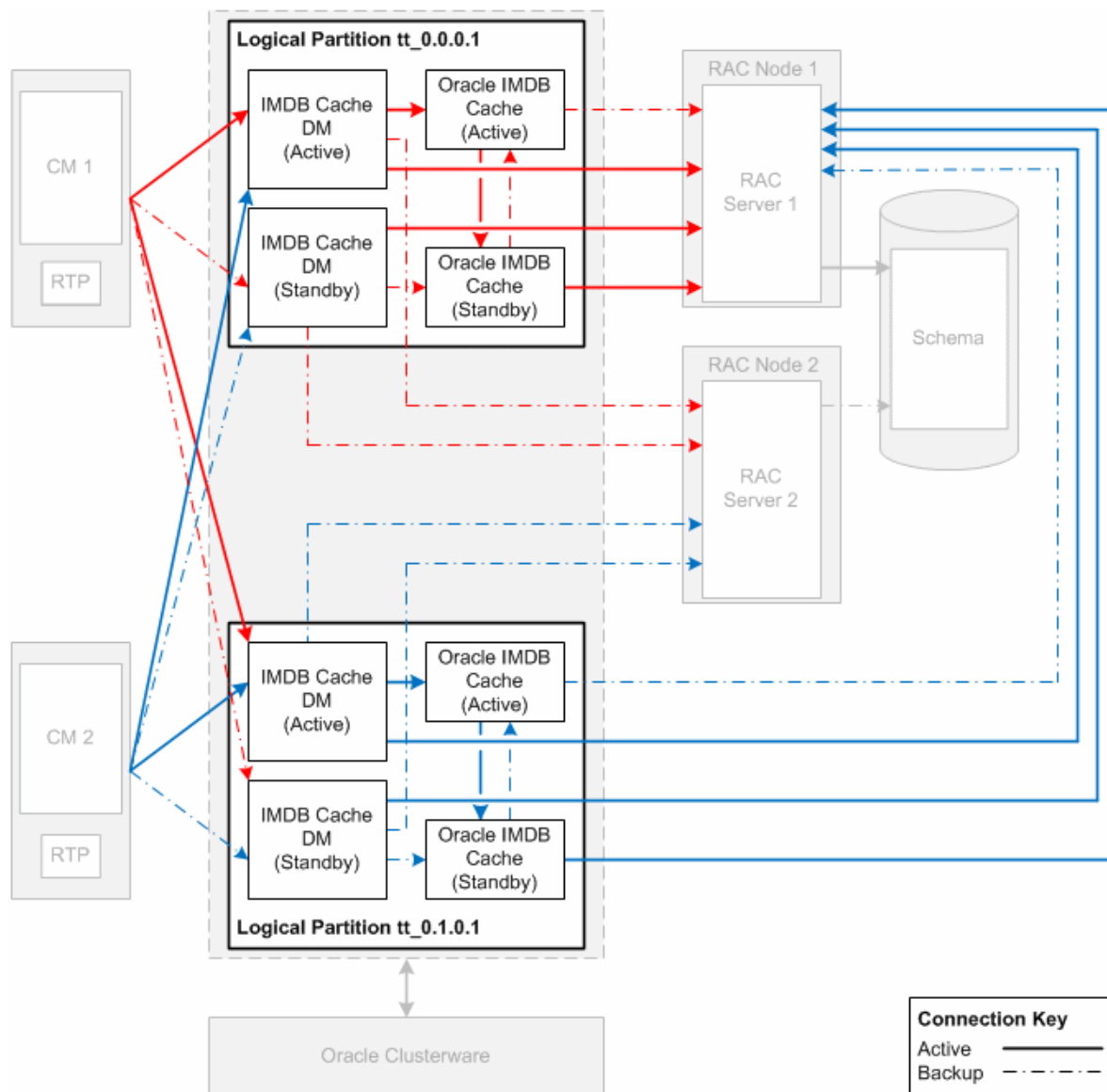
Configuring IMDB Cache Manager for High Availability

This section provides information about configuring IMDB Cache Manager for high availability.

The basic BRM high-availability architecture has one pair of *active* and *standby* IMDB Cache DM instances and one pair of *active* and *standby* Oracle IMDB Cache instances (data stores) for each BRM database schema. Larger high-availability systems can have several IMDB Cache DMs and Oracle IMDB Cache pairs for each schema.

For more information, see ["About IMDB Cache DMs and Data Stores in a High-Availability System"](#).

[Figure 13–2](#) shows a basic configuration of IMDB Cache DMs and their data stores in a high-availability system with two logical partitions and one BRM database schema. Dashed lines represent backup connections.

Figure 13–2 Basic IMDB Cache DM and Data Store Configuration for High Availability

Note: In a high-availability system, an IMDB Cache DM and its associated data store reside on the same physical server (*node*). Each node should contain only one DM–data store pair.

This section explains how to set up a high-availability BRM system that contains the following components:

- Two logical partitions for a single-schema database
- The **ttGrid** cache grid
- Data stores **tt_0.0.0.1** and **tt_0.1.0.1**

For more information about installing IMDB Cache Manager, including hardware and software requirements, see ["Installing IMDB Cache Manager"](#).

For an overview of IMDB Cache Manager, including information about cache grids and logical partitions, see ["Using Oracle IMDB Cache Manager"](#).

To set up a high-availability system for data stores **tt_0.0.0.1** and **tt_0.1.0.1**:

1. Install and configure the BRM database with Oracle RAC. See ["Configuring the BRM Database for High Availability"](#).
2. Install Oracle Clusterware. See the Oracle Clusterware documentation.
3. On each node on which you plan to configure a data store, install an instance of Oracle IMDB Cache. See the *Oracle TimesTen In-Memory Database Installation Guide*.

To support the example data stores **tt_0.0.0.1** and **tt_0.1.0.1**, you must configure a total of *four* instances of Oracle IMDB Cache (that is, an active and a standby instance for each data store). Each instance must reside on a different node.

Important: The primary group of the Oracle IMDB Cache owner should be the same as the primary group of the Oracle Clusterware owner. Their user names, however, can be different.

4. Install BRM. See "Installing BRM" in *BRM Installation Guide*.
5. On each node on which you plan to configure a data store, install an instance of the IMDB Cache DM. See ["Installing IMDB Cache Manager"](#).
6. Install any optional components that you want to add to your system.
7. Create and configure the active data stores **tt_0.0.0.1** and **tt_0.1.0.1**. See ["Creating and Configuring Active Data Stores"](#).
8. If you have existing BRM data that was created in a BRM system before IMDB Cache Manager was installed, run the **load_pin_uniqueness** utility to prepare the data for migration to an IMDB Cache Manager-enabled system.

Note: Stop and restart the CM and DM.

9. Create the schema and load BRM objects into the active data stores. See ["Initializing an Active Data Store"](#).
10. Configure clusters for the active data stores. See ["Configuring the Cluster for an Active Data Store"](#).
11. Configure the standby data stores. See ["Configuring Standby Data Stores"](#).
12. Create the active/standby data store pairs, and register them with Oracle Clusterware. See ["Creating Active and Standby Data Store Pairs"](#).
13. Configure IMDB Cache DM instances to connect to the active and standby data stores. See ["Associating IMDB Cache DM Instances with Data Stores"](#).
14. Configure the CMs to connect to the IMDB Cache DM instances. See ["Configuring Connection Managers for High Availability"](#).

Creating and Configuring Active Data Stores

To create and configure data stores for high availability, perform the following procedure on *each* node on which you want an active data store to reside.

1. Log on to the node for the active data store.

2. Create a directory for storing database files:

```
mkdir BRM_home/Database_Files_Location
```

For example:

```
mkdir BRM_home/database_files
```

Note: Oracle recommends using a local disk for database files instead of a network-mounted disk.

3. Go to the following directory:

```
cd IMDB_home/info
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

4. Add the data store attributes to the **sys.odbci.ini** data store configuration file (see ["sys.odbci.ini configuration file"](#)).

Note: You can edit the **sys.odbci.ini** file in the *IMDB_home/info* directory by commenting out the default configurations.

For example:

```
[DSN]
DataStore=BRM_home/brm_database_files/Data_Store_Name
OracleNetServiceName=PinDB
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
PLSQL=1
OracleNetServiceName=pin_db
oraclepwd=pin01
Driver=IMDB_homehome/lib/libtten.so
#Shared-memory size in megabytes allocated for the data store.
PermSize= 32
#Shared-memory size in megabytes allocated for temporary data partition,
generally half the size of PermSize.
TempSize=16
PassThrough=0
#Use large log buffer, log file sizes
LogFileSize=512
#Async repl flushes to disk before sending batches so this makes it faster
#on Linux
LogFlushMethod=2
#Limit Ckpt rate to 10 mb/s
CkptFrequency=200
CkptLogVolume=0
CkptRate=10
Connections=200
#Oracle recommends setting LockWait to 30 seconds.
LockWait=30
DurableCommits=0
CacheGridEnable=1
```

where:

- *DSN* is the data source name, which is the same as the data store name. The DSN must also be the same as the database alias name in the **tnsnames.ora** file. For more information about setting database alias names, see ["Making a Data Store Accessible to IMDB Cache DM"](#).
 - *BRM_home* is the directory in which BRM is installed.
 - *Data_Store_Name* is the name of the data store.
 - *IMDB_home* is the directory in which Oracle IMDB Cache is installed.
5. Save and close the file.
 6. Go to the *IMDB_home* directory and source the **ttenv.csh** file:

```
cd IMDB_home/bin
source ttenv.csh
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

7. Set up the Oracle IMDB Cache grid privileges in the BRM database:
 - a. Connect to the BRM database as a system administrator:

```
cd IMDB_home/oraclescripts
sqlplus sys as sysdba
```

- b. Run the following SQL scripts:

```
@IMDB_home/oraclescripts/initCacheGlobalSchema.sql "Data_Store_User";
@IMDB_home/oraclescripts/grantCacheAdminPrivileges.sql "Data_Store_User"
```

where *Data_Store_User* is the IMDB Cache data store user.

- c. Run the following commands to grant privileges:

```
grant all on TIMESTEN.TT_GRIDID to "Oracle_DB_User";
grant all on TIMESTEN.TT_GRIDINFO to "Oracle_DB_User";
```

where *Oracle_DB_User* is the BRM database user.

For more information, see the *Oracle TimesTen In-Memory Database Cache User's Guide*.

8. Create the data store:

```
cd IMDB_home/bin
ttIsql Data_Store_Name
```

where *Data_Store_Name* is the name of the data store, such as **tt_0.0.0.1** and **tt_0.1.0.1**.

9. Create the data store user and grant all permissions:

```
ttIsql Data_Store_Name
create user Data_Store_User identified by Data_Store_Password;
grant all to Data_Store_User;
```

where:

Data_Store_Name is the name of the data store.

Data_Store_User is the IMDB Cache data store user.

Data_Store_Password is the password for the IMDB Cache data store user.

Important: The IMDB Cache data store user must be the same as the BRM database user. However, the data store user password can be different from the database user password.

10. Set the data store user and password, and make the data store grid-enabled:

```
ttIsql "uid=Data_Store_User;pwd=Data_Store_Password;dsn=Data_Store_Name"
call ttcacheuidpwdset('Cache_Admin_User', 'Cache_Admin_User_Pwd');
call ttGridCreate('ttGrid');
call ttGridNameSet('ttGrid');
```

where:

Data_Store_User is the Oracle IMDB Cache data store user name, which must be the same as the BRM database user name.

Data_Store_Password is the password for the Oracle IMDB Cache user name.

Data_Store_Name is the name of the data store.

Cache_Admin_User and *Cache_Admin_User_Pwd* are the cache user name and password.

Important: Run the `call ttGridCreate('ttGrid')` command only once per database schema. For example, if you are configuring multiple active data stores for a single schema, run this command only when you configure the first active data store.

For more information about creating a cache grid, see the *Oracle TimesTen In-Memory Database Cache User's Guide*.

Note: To initialize the data stores for a multischema setup, you must generate the schema and load SQL files for each database schema by using the `pin_tt_schema_gen` utility. Then follow the preceding steps to initialize the data stores for each schema. See ["Generating the BRM Cache Group Schema"](#).

Initializing an Active Data Store

To load BRM objects into an active data store, you must perform the following procedures:

1. Generate the BRM cache groups schema using the BRM database, and extract the data from the BRM database for caching. See ["Generating the Schema and Load SQL Files for the Active Data Store"](#).
2. Create and initialize the BRM cache groups schema in the active data store. See ["Initializing the Data Store"](#).

Note: If you have existing BRM data that was created on a BRM system before IMDB Cache Manager was installed, you must run the `load_pin_uniqueness` utility *before* performing these procedures.

Generating the Schema and Load SQL Files for the Active Data Store

Use the **pin_tt_schema_gen** utility to generate the schema SQL file with the BRM cache groups schema and the load SQL file with the BRM data. For more information, see ["Generating the BRM Cache Group Schema"](#).

To generate the schema and load SQL files, perform the following procedure on the node on which the active data store resides:

1. Open the *BRM_home/bin/pin_tt_schema_gen.values* file, and configure the values in the file.

Note: You must generate the load SQL for each active data store.

For example, generate **tt_load_0.0.0.1.sql** with **\$db_no_for_load_sql** set to **0.0.0.1**, and generate **tt_load_0.1.0.1.sql** with **\$db_no_for_load_sql** set to **0.1.0.1** in the **pin_tt_schema_gen.values** file.

See ["Configuring the pin_tt_schema_gen.values File"](#).

2. Save and close the file.
3. Run the following command:
4. Run the **pin_tt_schema_gen** utility with the **-a** parameter:

```
source BRM_home/source.me.csh
```

```
./pin_tt_schema_gen -a
```

See ["pin_tt_schema_gen"](#).

Note: If you do not specify the values for **MAIN_DB{'user'}** and **MAIN_DB{'password'}** in the **pin_tt_schema_gen.values** file, the **pin_tt_schema_gen** utility prompts you to enter those values.

This updates the BRM database with unique indexes and non-null constraints and generates the following files:

- **tt_schema.sql**
- **tt_load_Logical_Partition.sql**
- **tt_drop.sql**

where *Logical_Partition* is the name of the logical partition in which the data store resides, such as **0.0.0.1** and **0.1.0.1**.

Initializing the Data Store

Use the schema and load SQL files to create the BRM cache groups schema and to load the BRM data.

To initialize an active data store, perform the following procedure on the node on which the active data store resides:

1. Set the data store user and password:

```
ttIsql "uid=Data_Store_User; pwd=Data_Store_Password; dsn=Data_Store_Name"
call ttcacheuidpwdset('Cache_Admin_User','Cache_Admin_User_Pwd');
```

where:

Data_Store_User is the Oracle IMDB Cache data store user name, which must be the same as the BRM database user name.

Data_Store_Password is the password for the Oracle IMDB Cache user name.

Data_Store_Name is the name of the data store, such as **tt_0.0.0.1** and **tt_0.1.0.1**.

Cache_Admin_User and *Cache_Admin_User_Pwd* are the cache user name and password.

2. Start the cache agent:

```
call ttcachestart;
```

3. Create the schema:

```
run BRM_home/bin/tt_schema.sql;
```

4. Create stored procedures:

```
run BRM_home/sys/dm_tt/data/tt_create_pkg_pin_sequence.plb;
```

```
run BRM_home/sys/dm_tt/data/tt_create_procedures.plb;
```

```
run BRM_home/sys/dm_tt/data/create_tt_wrappers.plb;
```

Note: Load the stored procedures in **tt_create_pkg_pin_sequence.plb** before the procedures in **tt_create_procedures.plb**.

Configuring the Cluster for an Active Data Store

To set up the cluster configuration for an active data store:

1. Log on to the node on which the active data store resides.
2. Go to the following directory:

```
cd IMDB_home/info
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

3. Add the following entries to the **cluster.oracle.ini** data store configuration file.

You can edit the **cluster.oracle.ini** file in the *IMDB_home/info* directory by commenting out the default configurations.

Note: The **MasterHosts** entry identifies the nodes on which a pair of active/standby data stores resides. The order in which the nodes are specified sets the default states (active, standby) of the data stores.

```
[DSN]
```

```
MasterHosts = Active_Node, Standby_Node
```

```
ScriptInstallDir = /export/home/ttinstaller/TimesTen/tt1121_HA/info/crs_scripts
```

```
CacheConnect = Y
```

```
ReturnServiceAttribute = RETURN_TWOSAFE
```

```
GridPort = Active_Port, Standby_Port
```

```
AppName = DataStoreName
```

```
AppType = Active
```

```
AppCheckCmd = BRM_home/bin/pin_ctl_dmtt.sh check
```

```
AppStartCmd = BRM_home/bin/pin_ctl_dmtt.sh start
```

```
AppStopCmd = BRM_home/bin/pin_ctl_dmtt.sh stop
MonInterval = MonitorInterval
AppRestartAttempts = NumAttempts
AppUptimeThreshold = AttemptReset
```

where:

- DSN is the data source name, which is the same as the data store name. The DSN must also be the same as the database alias name in the **tnsnames.ora** file. For more information about setting database alias names, see ["Making a Data Store Accessible to IMDB Cache DM"](#).
- *Active_Node* is the host name for the active data store.
- *Standby_Node* is the host name for the standby data store.
- *Active_Port* is an unused port number assigned to the active data store.
- *Standby_Port* is an unused port number assigned to the standby data store.
- *DataStoreName* is the name of the active data store.
- *MonitorInterval* specifies the amount of time, in seconds, that the Oracle Clusterware processes monitor an active/standby pair.
- *NumAttempts* specifies the number of times Oracle Clusterware attempts to connect to the IMDB Cache data store before the IMDB Cache data store fails over. When set to **0**, the IMDB Cache data store fails over immediately.
- *AttemptReset* specifies the amount of time, in seconds, until the **AppRestartAttempts** entry is reset.

Notes:

- *Active_Port* and *Standby_Port* are ports on the private network that is set up as part of Oracle Clusterware installation.
 - The **AppName** and **AppType** entries must be the same on both nodes.
 - The **cluster.oracle.ini** data store configuration file includes other optional entries that can be used to fine-tune your system. For more information, see the *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.
-

4. Save and close the file.

Configuring Standby Data Stores

To configure a standby data store for an active data store, perform this procedure on the node on which you want the standby data store to reside.

Before performing this procedure, obtain the following information:

- The name and location of the directory used to store database files for the active data store. See step 2 in ["Creating and Configuring Active Data Stores"](#).
- A copy of the entries added to the **sys.odbci.ini** file for the active data store. See step 4 in ["Creating and Configuring Active Data Stores"](#).
- A copy of the entries added to the **cluster.oracle.ini** file for the active data store. See step 3 in ["Configuring the Cluster for an Active Data Store"](#).

To create and configure a standby data store for an active data store:

1. Log on to the node on which you want the standby data store to reside.
2. Create a directory for storing database files:

```
mkdir BRM_home/Database_Files_Location
```

Important: The name and location of the directory must match the corresponding directory created on the node on which the active data store resides. See step 2 in ["Creating and Configuring Active Data Stores"](#).

3. Go to the following directory:

```
cd IMDB_home/info
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

4. In the **sys.odbci.ini** data store configuration file, add the *same* entries that you added to the active data store's **sys.odbci.ini** file. See step 4 in ["Creating and Configuring Active Data Stores"](#).
5. Save and close the file.
6. In the **cluster.oracle.ini** data store configuration file, add the *same* entries that you added to the active data store's **cluster.oracle.ini** file. See step 3 in ["Configuring the Cluster for an Active Data Store"](#).
7. Save and close the file.

Creating Active and Standby Data Store Pairs

Use this procedure to perform the following tasks:

- Create a pair of active and standby data stores.
- Register the TimesTen agent and the data stores with Oracle Clusterware.
- Replicate the active data store's BRM cache groups schema and data in the standby data store.

Perform the procedure on each active data store node.

1. Go to the **bin** directory on the node on which the active data store resides:

```
cd IMDB_home/bin
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

2. Register the cluster information on the host as a root user by entering the following command:

```
ttCWAdmin -ocrconfig
```

3. Start the Oracle IMDB Cache cluster agent using the TimesTen administrator user login:

```
ttCWAdmin -init
```

4. Create an active/standby pair replication scheme:

```
ttCWAdmin -create -dsn Data_Store_Name
```

where *Data_Store_Name* is the name of the data store, such as **tt_0.0.0.1** or **tt_0.1.0.1**.

5. Provide the required information, such as the admin user ID and password. A confirmation message is displayed when the registration is complete.
6. Start the active/standby pair replication scheme:

```
ttCWAdmin -start -dsn Data_Store_Name
```

Oracle Clusterware automatically starts the data stores.

7. Load data into the active/standby pair replication scheme:

```
ttisql Data_Store_Name  
run BRM_home/bin/tt_load_Logical_Partition.sql;
```

where *Logical_Partition* is the database number of the logical partition in which the data stores reside, such as **0.0.0.1** or **0.1.0.1**.

To initialize the data stores for a multischema setup, you must generate the schema and load SQL files for each database schema by using the **pin_tt_schema_gen** utility. Then follow the steps in this section to initialize the data stores for each schema.

For more information about registering data stores with Oracle Clusterware, see *Oracle Clusterware Administration and Deployment Guide*.

For more information about TimesTen replication, see *TimesTen to TimesTen Replication Guide*.

Associating IMDB Cache DM Instances with Data Stores

For high availability, you need an active and a standby instance of each IMDB Cache DM. The active DM instance is connected to the active data store, and the standby DM instance is connected to the standby data store.

To connect active and standby IMDB Cache DM instances to the active and standby data stores, perform the following procedures on each node on which an active or a standby data store resides:

1. [Making a Data Store Accessible to IMDB Cache DM](#)
2. [Configuring an IMDB Cache DM Instance for a Data Store](#)
3. [Configuring pin_ctl to Start and Stop IMDB Cache DM](#)

Note: These procedures assume that you have installed an instance of IMDB Cache Manager on each node on which an active or a standby data store resides.

Making a Data Store Accessible to IMDB Cache DM

To configure a data store so that IMDB Cache DM can directly connect to it, perform the following procedure on the node on which the data store resides:

1. Open the **tnsnames.ora** file located in the directory specified by **\$TNS_ADMIN**.
2. Add the following entry:

```
Database_Alias_Name = (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT =  
))  
                        (CONNECT_DATA =
```

```
(SERVICE_NAME = Data_Store_Name)
(SERVER = timesten_direct )))
```

where:

- *Database_Alias_Name* is the data store name specified in the **sys.odbci.ini** file.
- *Data_Store_Name* is the data store name specified in the **sys.odbci.ini** file.
- **timesten_direct** indicates that instances of IMDB Cache DM on the same node as the data store can directly connect to the data store.

Note: You must add a separate entry for each logical partition.

For example:

```
tt_0.0.0.1 = (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT = ))
              (CONNECT_DATA =
                (SERVICE_NAME = tt_0.0.0.1)
                (SERVER = timesten_direct )))

tt_0.1.0.1 = (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT = ))
              (CONNECT_DATA =
                (SERVICE_NAME = tt_0.1.0.1)
                (SERVER = timesten_direct )))
```

3. Save and close the file.

Configuring an IMDB Cache DM Instance for a Data Store

Use the existing IMDB Cache DM installation to configure the settings for the active data store.

To configure IMDB Cache DM for the active data store:

1. Open the *BRM_home/sys/dm_tt/pin.conf* file.

2. Set the **tt_ha_enabled** entry to 1:

```
- dm tt_ha_enabled 1
```

3. Set the **sm_database_tt** entry to the data store:

```
- dm sm_database_tt Data_Store_Name
```

where *Data_Store_Name* is the name of the data store, such as **tt_0.0.0.1** or **tt_0.1.0.1**.

4. Set the **sm_pw_tt** entry to the data store password:

```
- dm sm_pw_tt Data_Store_Password
```

Data_Store_Password is the password for the IMDB Cache data store user.

Important: The IMDB Cache data store user must be the same as the BRM database user. However, the data store user password can be different from the database user password.

5. Set the **logical_partition** entry to 1 to enable logical partitioning:

```
- dm logical_partition 1
```

6. Set the **pcm_op_max_retries** entry to a value of 1 or greater. This entry specifies the number of times IMDB Cache DM attempts to connect to an opcode after a failure.

```
- dm pcm_op_max_retries 3
```

7. Set the **pcm_reconnect_max_retries** entry to a value of 1 or greater.

```
- dm pcm_reconnect_max_retries 3
```

8. Save and close the file.

Configuring pin_ctl to Start and Stop IMDB Cache DM

To configure the **pin_ctl** utility to start and stop an instance of IMDB Cache DM, perform the following procedure on the node on which the DM and its associated data store reside:

1. Set the **TIMESTEN_HOME** environment variable to the directory in which the IMDB Cache DM is installed on the node. For example:

```
/export/home/ttinstaller/TimesTen/tt1121_HA
```

2. Go to the directory in which the IMDB Cache DM is installed and source the **source.me** file:

- Bourne shell:

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

- C shell:

```
source source.me.csh
```

3. Open the *BRM_home/bin/pin_ctl.conf* file in a text editor.
4. Add the following line to the startup configuration section of the file:

```
Start_DMTTInstance_Service_Name clpidproc:DMTTInstance_Service_Name:  
cport:DMTT_Port host:DMTT_Host dbno:Data_Store_DB_Number
```

where:

- *Start_DMTTInstance_Service_Name* is the name of the start command for the IMDB Cache DM instance.
- *DMTTInstance_Service_Name* is a simple process name matching filter.
- *DMTT_Port* is the IMDB Cache DM port number.
- *DMTT_Host* is the IMDB Cache DM host name.
- *Data_Store_DB_Number* is the data store database number.

For example:

```
start_dm_tt cpidproc:dm_tt: cport:1234 host:vm31230 dbno:0.0.0.1
```

5. Save and close the file.
6. To ensure that **pin_ctl** is configured correctly, run the following commands:

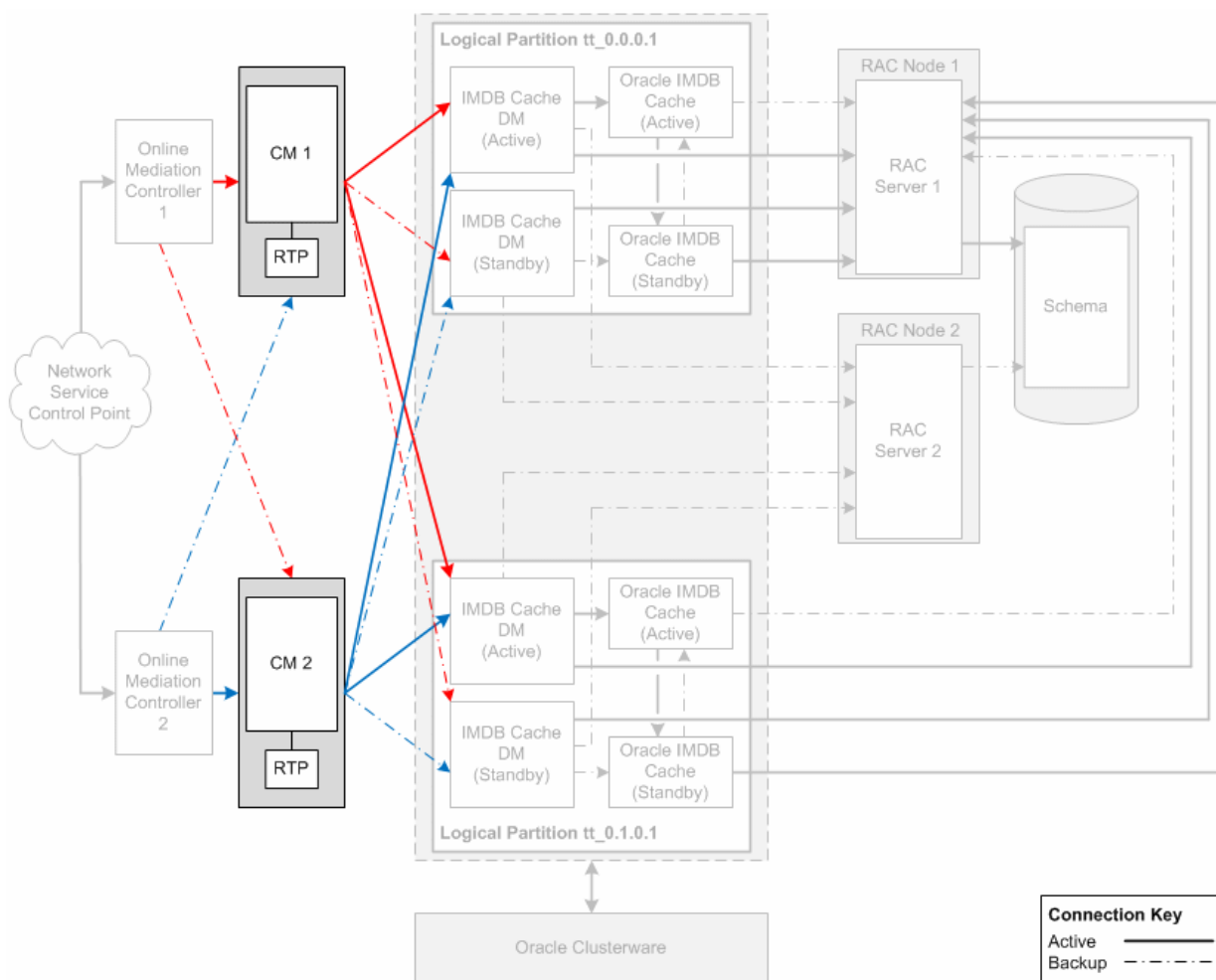
- **pin_ctl start dm_tt**
- **pin_ctl stop dm_tt**

For more information about configuring `pin_ctl` for high availability, see ["Using the pin_ctl Utility to Monitor BRM"](#).

Configuring Connection Managers for High Availability

Figure 13–3 shows a basic configuration of CMs in a high-availability system with two logical partitions. Dashed lines represent backup connections.

Figure 13–3 Basic CM Configuration for High Availability



To configure CMs for high availability:

1. Install at least two CMs. Use as many as you estimate will support your workload, and then add one more.
2. Install one real-time pipeline for each CM. See ["Configuring Pipeline Manager"](#).
3. Connect each CM to all the active and standby IMDB Cache DMs in your BRM System. See ["Connecting CMs to IMDB Cache DMs"](#).

Connecting CMs to IMDB Cache DMs

To configure a CM to connect to the active and standby IMDB Cache DM instances in a high-availability system, set the CM configuration file (`BRM_home/sys/cm/pin.conf`) entries shown in [Table 13–3](#). These settings minimize CM request failures and enable

CM connections to succeed if an IMDB Cache DM failover occurs.

For information about how CMs handle IMDB Cache DM failure, see ["About Connection Managers in a High-Availability System"](#).

Table 13–3 CM pin.conf Entries for a High-Availability System

Configuration Entry	Description
dm_pointer	<p>Specifies the host and port number of the IMDB Cache DM instances to connect to. Include an entry for each active and standby pair of IMDB Cache DMs in your system. Oracle recommends that the active DM be listed first in each entry.</p> <p>The CM pin.conf file should contain one dm_pointer entry <i>per logical partition</i>. Therefore, because the active and standby DMs in each pair support the same logical partition, they must be on the <i>same dm_pointer</i> line:</p> <pre>- cm dm_pointer lp_number ip active_dm_host active_dm_port ip standby_dm_host standby_dm_port</pre> <p>Example</p> <pre>- cm dm_pointer 0.0.0.1 ip 156.151.2.168 33950 ip 168.35.37.128 12960 - cm dm_pointer 0.1.0.1 ip 156.151.2.168 32250 ip 168.35.37.128 12850</pre>
pcm_timeout_in_msecs	<p>Specifies the amount of time in milliseconds that the CM waits for a response from an IMDB Cache DM before failing over to the standby DM. This entry is called the CM's long (failover) timeout (see "Configuring Multilevel CM Timeout for Client Requests").</p> <p>The default value is 120000 (120 seconds). For high-availability systems, the recommended value is 100000 (100 seconds).</p> <p>Important: In a high-availability system, each client-side component should have a longer timeout period than the server-side component that it calls. For example, to minimize CM request failures, make this timeout long enough for Oracle Clusterware to restart the IMDB Cache DM. For suggested timeout values, see "About Setting Up a High-Availability System".</p> <p>Example</p> <pre>- cm pcm_timeout_in_msecs 100000</pre>
pcm_op_max_retries	<p>Specifies the maximum number of times an opcode is retried in the Portal Communications Model (PCM). The default value is 1. For high availability, the value must be at least 2.</p> <p>Example</p> <pre>-cm pcm_op_max_retries 2</pre>
cm_op_max_retries	<p>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.</p> <p>Example</p> <pre>- cm cm_op_max_retries 2</pre>
pcm_bad_connection_retry_delay_time_in_secs	<p>Specifies the interval, in seconds, at which the CM tries to connect to the active IMDB Cache DM after it fails to connect. See "How CMs Handle IMDB Cache DM Failure".</p> <p>Important: In a high-availability system, each client-side component should have a longer timeout period than the server-side component that it calls. For suggested timeout values, see "About Setting Up a High-Availability System".</p> <p>Example</p> <pre>-cm pcm_bad_connection_retry_delay_time_in_secs 100</pre> <p>Note: This entry appears in both the CM and IMDB Cache DM pin.conf files.</p>

Restoring a High-availability System after Failover

After failed components in a high-availability system are fixed, return the system to its original configuration by switching the workload back to the primary components. This enables the system to use its optimal architecture.

The following sections explain how to restore components in a high-availability system:

- [Switching Back to the Primary Oracle RAC Instance](#)
- [Ensuring All Accounts Are Billed](#)

Switching Back to the Primary Oracle RAC Instance

To restart a failed Oracle RAC instance, run the following command:

```
srvctl start instance -d racDatabaseName -i primary_racInstanceName
```

where:

- *racDatabaseName* is the name of the Oracle database.
- *primary_racInstanceName* is the name of the primary (preferred) Oracle RAC instance.

For information about the **srvctl** command, see the Oracle RAC documentation.

After a failed database instance is restarted, the services for which it is the primary database instance do not automatically switch back to it from the backup instance.

To switch a database service from its backup database instance to its primary database instance, run the following command:

```
srvctl relocate service -d racDatabaseName -s serviceName  
-i backup_racInstanceName -t primary_racInstanceName -f
```

where:

- *racDatabaseName* is the name of the Oracle database.
- *serviceName* is the name of the database service that the primary Oracle RAC instance originally supported.
- *backup_racInstanceName* is the name of the backup (available) Oracle RAC instance.
- *primary_racInstanceName* is the name of the primary (preferred) Oracle RAC instance.

Note: Switching database services back to their primary Oracle RAC instance causes a service interruption. Usually, however, switching back to the primary node takes less time than failing over to the backup node.

Ensuring All Accounts Are Billed

If a system failure occurs while the **pin_bill_day** application is running, some operations might fail, and thus bills might not be generated for all accounts.

To ensure that all accounts are billed, Oracle recommends rerunning **pin_bill_day** after the system is restored.

Part III

Improving Performance

Part III describes how to improve performance for an Oracle Communications Business and Revenue Management (BRM) system. It contains the following chapters:

- [Improving BRM Performance](#)
- [Optimizing Pipeline Manager Performance](#)
- [Optimizing BRM for Prepaid and Postpaid Convergence](#)

Improving BRM Performance

This chapter provides information on evaluating and improving the performance of your Oracle Communications Billing and Revenue Management (BRM) system.

Before you read this chapter, you should be familiar with BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.

For information on troubleshooting BRM, see ["Resolving Problems in Your BRM System"](#).

For information on tuning Pipeline Manager, see ["Optimizing Pipeline Manager Performance"](#).

Monitoring Performance

You can use BRM diagnostics tools to monitor performance. For example:

- You can monitor opcode latency.
- You can monitor event data record (EDR) throughput in a pipeline.

See ["About Monitoring BRM"](#).

Improving Connection Manager Performance

For information about Connection Managers (CMs), see "The Business Process Tier" in *BRM Concepts*.

To improve CM performance, see the following topics:

- [Increasing CM Login Performance](#)
- [Load Balancing CMs](#)
- [Improving Performance for Loading Large Price Lists](#)
- [Improving Real-Time Rating Performance](#)
- [Improving the Performance of your Multithreaded Applications](#)
- [Logging Noncurrency Events](#)
- [Specifying the Number of Connections to CMs](#)
- [Setting the CM Time Interval between Opcode Requests](#)

Increasing CM Login Performance

To increase CM login performance, see the following topics:

- [Using CM Proxy to Allow Unauthenticated Log On](#)
- [Turning Off Session-Event Logging](#)
- [Turning Off the Checking of Logons and Passwords](#)

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.

Caution: 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 ["Locations of Configuration and Properties 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 storable object for each client connection, but you can suppress the creation of these session storable 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. See ["Starting and Stopping the BRM System"](#).

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, or HP-UX IA64; or **a** for AIX. For example:

```
- cm    cm_login_module    ./cm_login_null.so
```

3. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).

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 Processes"](#).
- 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
```

Tip: 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
```

Improving Performance for Loading Large Price Lists

If you have a large price list, you can improve performance in the following ways:

- Cache pricing data, such as G/L IDs and resource IDs. In a test environment where you modify your price list often, caching pricing data improves performance because there is no need to load price reference objects every time you commit the price list to the database.

Important: 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 price list, 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 the price list.

- Turn off event logging for price list creation events.

Note: When you turn off event logging, BRM still stores audit trail information for products, deals, and plans; however, there will not be an event log of when the price plans were modified and who modified them.

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
```

Important: For policy-driven charging, the **fm_offer_profile_cache** entry must be present in the **pin.conf** file.

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

7. Save the file.
8. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).

Improving Real-Time Rating Performance

You can improve real-time rating performance by doing the following:

- [Changing the Precision of Rounded and Calculated Values](#)

- [Setting the Interval for Checking for Price List Changes](#)
- [Setting the Interval for Updating Zone Maps](#)
- [Filtering the ERAs Considered during Rating and Discounting](#)
- [Enabling and Disabling the Caching of Customized Products](#)
- [Configuring the Maximum Number of Products and Discounts Cached](#)

Changing the Precision of Rounded and Calculated Values

To improve performance, you can change the precision of rounded values and of values calculated by real-time 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.

Important: You must stop and restart the CM after you change these values. See ["Starting and Stopping the BRM System"](#).

Setting the Interval for Checking for Price List Changes

You can set the interval at which BRM checks for changes to the price list. If you change the price list frequently, you may want to use a shorter interval. If your price list is 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. See ["Starting and Stopping the BRM System"](#).

Setting the Interval for Updating Zone Maps

To specify how frequently BRM checks for changes to zone 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. See ["Starting and Stopping the BRM System"](#).

Filtering the ERAs Considered during Rating and Discounting

By default, real-time 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 real-time 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 modifying a field in the **rating** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To specify the ERA types:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsRating bus_params_rating.xml
```

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

2. Search the XML file for following line:

```
<EnableEras>serviceAndAccount</EnableEras>
```

3. 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.

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

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

```
pin_bus_params bus_params_rating.xml
```

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

7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser or by using the **robj** command with the **testnap** utility.

For more information, see the following topics in *BRM Developer's Guide*:

- Using testnap
 - Reading Objects by Using Object Browser
8. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).
 9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Enabling and Disabling the Caching of Customized Products

When you use advanced customization to create customized products, BRM uses the customized products for rating. You can control whether customized products are cached for use by the real-time rating engine.

- If you choose not to cache customized products (the default setting), the real-time rating engine retrieves customized product data from the database during rating. This slows rating performance but minimizes the memory impact of customized products.
- If you choose to cache customized products, the CM size grows as customized products are created. Because the products are cached in memory, however, rating performance is increased.

You enable product caching by changing the **EnableTailormadeCache** field in the **rating** instance of the **/config/business_params** object from **0** to **1**. You can disable caching by changing the field back to **0**.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To enable caching of customized products:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsRating bus_params_rating.xml
```

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

2. Search the XML file for following line:

```
<EnableTailormadeCache>0</EnableTailormadeCache>
```

3. Change **0** to **1**.

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

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.

6. Use the following command to load the change into the `/config/business_params` object:

```
pin_bus_params bus_params_rating.xml
```

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

7. To verify that all fields are correct, you can display the `/config/business_params` object by using Object Browser or by using the `robj` command with the `testnap` utility.

For information on using `testnap`, see "Using testnap" in *BRM Developer's Guide*.

For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

8. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).
9. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see ["pin_multidb"](#).

Configuring the Maximum Number of Products and Discounts Cached

Products and discounts that are used during the real-time 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 products and discounts that can be stored in the CM cache.

- When you set the maximum to a nonzero value, BRM prevents the real-time rating engine from storing more than the specified number of products and discounts in CM cache. When the maximum number is reached, BRM flushes 10% of the products and discounts from cache that have been used the least.

Note: The maximum number of products and discounts that should be stored in CM cache depends on the your business needs.

- When you set the maximum to zero, BRM does not regulate the number of products and discounts stored in the CM cache. This is the default.

You configure the maximum number of products and discounts that can be cached by configuring the `ProductsDiscountsThreshold` field in the `rating` instance of the `/config/business_params` object.

To set a maximum number of products and discounts that can be cached:

1. Use the following command to create an editable XML file from the `rating` instance of the `/config/business_params` object:

```
pin_bus_params -r BusParamsRating bus_params_rating.xml
```

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

2. Search the XML file for following line:

```
<ProductsDiscountsThreshold>0</ProductsDiscountsThreshold>
```

3. Change 0 to the maximum number of products and discounts that you would like stored in cache. The default value of 0 specifies to not regulate the number of products and discounts in the CM cache.

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

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

```
pin_bus_params bus_params_rating.xml
```

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

7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser or by using the **robj** command with the **testnap** utility.

For more information, see the following topics in *BRM Developer's Guide*:

- Using testnap
 - Reading Objects by Using Object Browser
8. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).
 9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Improving the Performance of your Multithreaded Applications

You can improve the performance of multithreaded applications by controlling the thread load and monitoring the thread activity.

Controlling Thread Load on Your Multithreaded Application

The number of child threads allowed is controlled by the **children** entry in the application **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**:

1. Go to the directory from which you run the MTA.
2. Start the **pin_mta_monitor** utility using 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 as the value for *number* in the appropriate command (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 Your Multithreaded Application

Use the **pin_mta_monitor** utility to monitor the thread activity of your MTA.

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 using 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
```

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:


```
- 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.

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 simultaneously. 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.

See ["Starting and Stopping the BRM System"](#).

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 3
```

The value of this entry specifies the time interval in minutes.

3. Save and close the file.
4. Stop and restart the CM. See ["Starting and Stopping the BRM System"](#).

Using Connection Manager Master Processes

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).

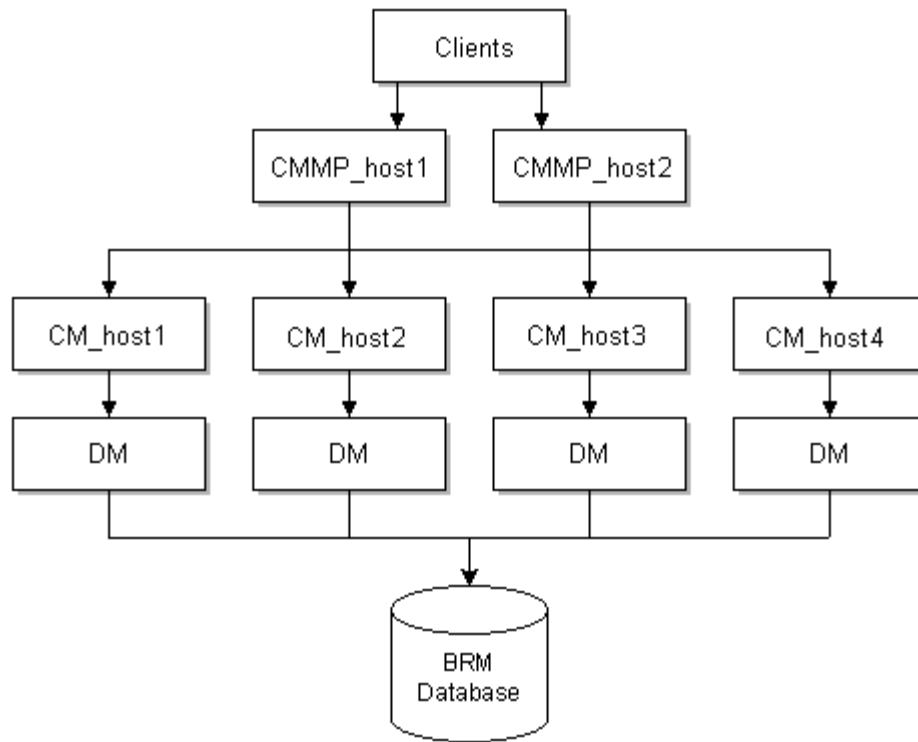
The CMMP does not validate whether the CMs it points to are running. If the CMMP sends a transaction to a nonfunctioning CM, the client can tell that the CM is offline and tries to connect to a different CM or CMMP. Therefore, you should include multiple **cm_ptr** entries to the same CMMP or to more 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 [Figure 14-1](#):

- 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 Data Manager (DM).

Figure 14–1 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. See ["Locations of Configuration and Properties Files"](#).
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.

Improving Data Manager and Queue Manager Performance

For information about Data Managers (DMs), see "The Data Management Tier" in *BRM Concepts*.

To improve DM performance, see the following topics:

- [About Queuing-Based Processes](#)
- [Configuring DM Front Ends and Back Ends](#)
- [Configuring Multiple Pipelines in the DM to Improve Performance](#)
- [Setting DM Shared Memory Size](#)
- [Reducing Resources Used for Search Queries](#)
- [Load Balancing DMs](#)
- [Optimizing Memory Allocation during Database Searches](#)
- [Improving BRM Performance during Database Searches](#)

- [Increasing DM CPU Usage](#)
- [Examples of DM Configurations](#)

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.

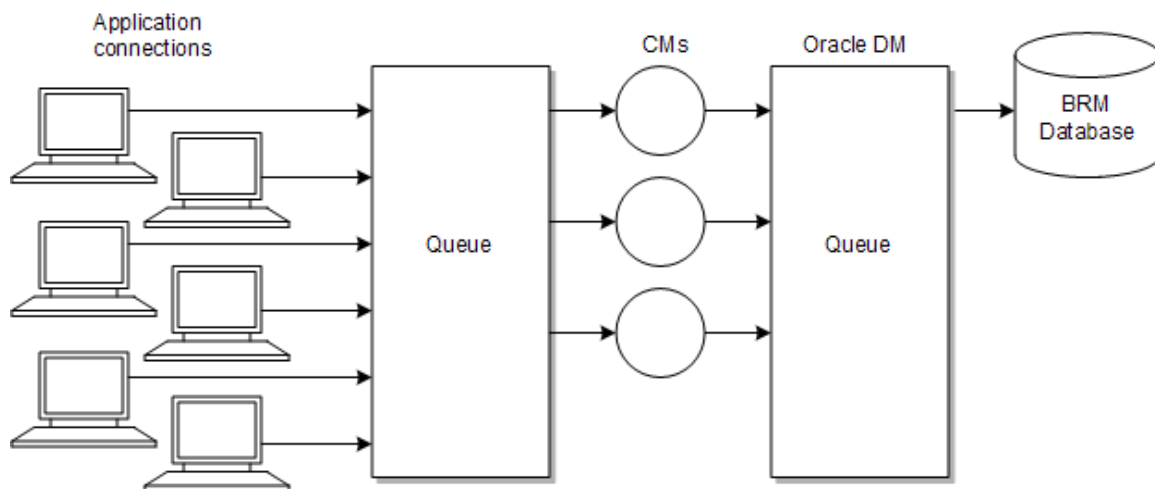
Queuing is used in two different types of system components:

- The RADIUS Manager, 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.

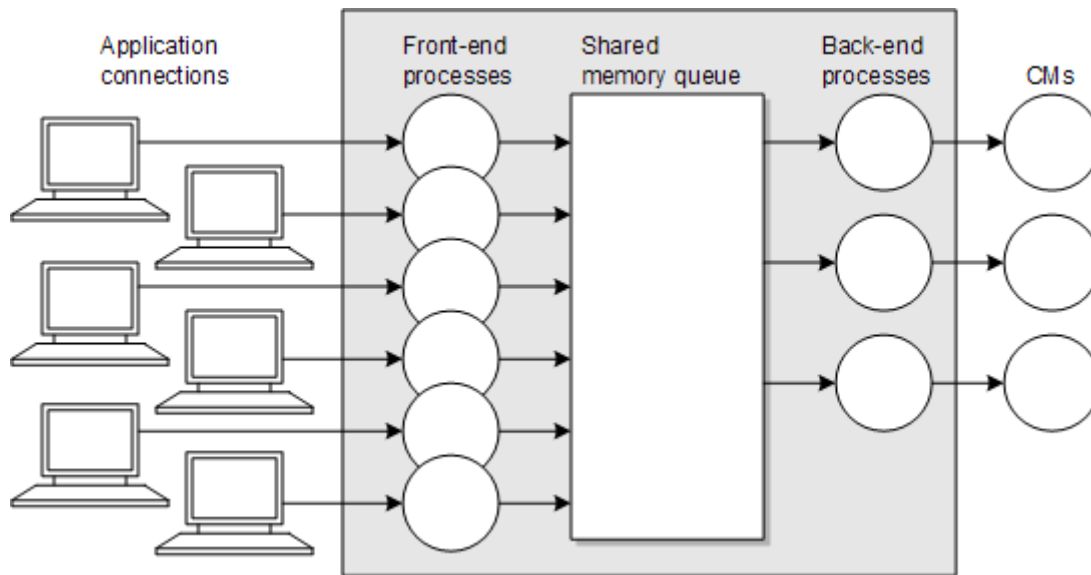
[Figure 14-2](#) shows an example of where queuing takes place in the BRM system architecture. Queuing occurs in two locations. In this example, connections are queued in the RADIUS Manager and the in the database DM.

Figure 14-2 BRM Queuing Locations



Example of Queuing in a Client-to-CM Connection

[Figure 14-3](#) 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 14–3 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 finish.
- **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. See ["Starting and Stopping the BRM System"](#).

Important: 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 master search thread.
- Two connections for each instance of Customer Center.

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 enables 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 master thread searches.

For example:

```
-dm dm_n_be 24  
-dm dm_trans_be_max 22
```

The example above enables 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 aborting 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 Finish

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 finish 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.

Configuring Multiple Pipelines in the DM to Improve Performance

By default, the front-end processes in the DMs write requests to a pipeline and the back-end processes listen to the pipeline and pick up the request. When all the back-end processes listen to a single pipeline, resources are wasted and performance might become slow. To improve performance, configure multiple pipelines in the DM, with each pipeline serving a set of back ends. The front-end processes send requests to the pipelines using the round-robin method. The back-end processes listen to the pipelines in order. For example, the first set of back-end processes listens to pipeline 1, the second set of processes listens to pipeline 2, and so on.

To configure multiple pipelines, include the following entry in the DM **pin.conf** file:

```
-dm dm_n_op_fifo nn
```

where *nn* is the number of pipelines. The number of your back-end processes, specified in the **dm_n_be** entry, must be a multiple of the pipelines you configure. The default is 1.

Tip: Start by configuring one pipeline for every six back-end processes and adjust the number of pipelines according to your requirements.

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) and must be a multiple of 8192.
- **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 **206158429184** bytes (192 GB). The value of **dm_bigsize** should always be set less than the value of **dm_shmsize** and must be a multiple of 1024.

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. See ["Starting and Stopping the BRM System"](#).

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 zone maps. Allocate 1 MB of memory in the **dm_bigsize** entry for every 3000 lines in a zone 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_bigszie 8388608
```

Note: The value for **dm_shmsize** must be a multiple of 1024. The value of **dm_bigszie** 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.

$100,000 \times 16 \times 2 = 3,200,000$ (3.2 MB), which is rounded up to a multiple of 8192. For example, **dm_bigszie** would be set to 3203072 or 391×8192 .

As a general rule, **dm_shmsize** should be approximately 4 to 6 times larger than **dm_bigszie**.

- Memory used by normal 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 executes $n+1$ 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 ($[25 \times 4] + 1$). The default setting is 80, indicating that BRM executes 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.

Important: 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. See ["Starting and Stopping the BRM System"](#).

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 14–1](#) 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 14–1 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

Table 14–1 (Cont.) Example 1 DM Configuration

Daemon/program	pin.conf entry	Value
dm_oracle	dm_trans_be_max	20
dm_oracle	dm_shmsize	67108864
dm_oracle	dm_bigszie	1048576

Example 2: BRM 36-CPU database server configuration

The example shown in [Table 14–2](#) shows a BRM system that uses:

- A 36x336 MHz CPU database server.
- Four 4x400 MHz CPU CM/DM/EM systems.

Table 14–2 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_bigszie	6291456

Improving Interprocess Communication (IPC) Performance

By default, CM and DM processes communicate through AF_INET sockets. You can increase your system's interprocess communication (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 14–3](#).

Table 14–3 IPC Socket Types

Socket type	Description
AF_UNIX	<p>Provides communication through a local socket file that the DM creates each time it starts.</p> <p>Note: If the DM finds a socket file when it starts, it deletes the existing file and creates a new one.</p> <p>These sockets provide the fastest IPC performance but can be used only by processes located on the same machine.</p>
AF_INET	<p>Provides communication through an IP address. These sockets are slower than AF_UNIX sockets, but they allow communication between processes on separate machines.</p>

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:

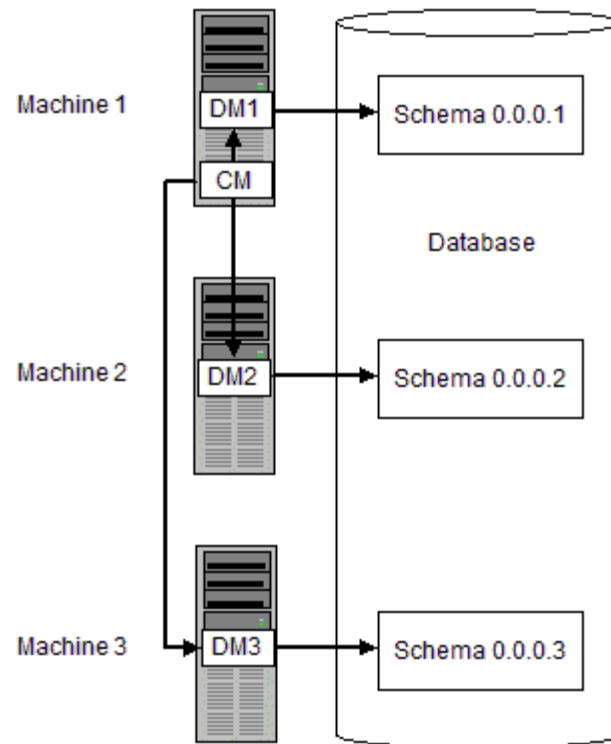
- **dm dm_port** *PortNumber*

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 14-4](#):

Figure 14–4 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

```

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 14–4 lists the features enabled by `/config/business_params` objects that most heavily impact performance.

Table 14–4 BRM Features that Heavily Impact Performance

Group	Name	Description	Default Value
activity	LightWeightAuthorization	Enables and disables lightweight authorization. See "Using Lightweight Authorization" in <i>BRM Telco Integration</i> .	Disabled
invoicing	SubARItemsIncluded	Indicates whether subordinate account invoices include A/R items. See "About Invoicing for Hierarchical Account Groups" in <i>BRM Designing and Generating Invoices</i> .	A/R items are included
billing	CacheResidencyDistinction	Enables and disables object cache residency distinction. See " About Convergent BRM Systems ".	Disabled
billing	EnableARA	Specifies whether Revenue Assurance is enabled for out-of-cycle billing. See "Configuring Bill Now, On-Demand Billing, and Auto-Triggered Billing to Collect Revenue Assurance Data" in <i>BRM Collecting Revenue Assurance Data</i> .	Disabled
billing	GeneralLedgerReporting	Enables and disables general ledger reporting. If disabled, the journal is not generated. See "Enabling and Disabling General Ledger Collection in BRM" in <i>BRM Collecting General Ledger Data</i> .	Enabled
rating	EnableEras	Specifies how to enable ERAs: <ul style="list-style-type: none"> 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 <i>BRM Managing Accounts Receivable</i> .	Disabled
rating	EnableTailormadeCache	Specifies if tailor-made products must be maintained in the rating cache. When enabled, tailor-made products are stored in cache. See " Enabling and Disabling the Caching of Customized Products ".	Disabled
multi-bal	BalanceMonitoring	Enables and disables balance monitoring. See "About Balance Monitoring" and "Enabling Balance Monitoring in BRM" in <i>BRM Managing Accounts Receivable</i> .	Disabled

Table 14–4 (Cont.) BRM Features that Heavily Impact Performance

Group	Name	Description	Default Value
multi-bal	LockConcurrency	Indicates the concurrency of object locking. Possible values are: <ul style="list-style-type: none"> 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 <i>BRM Developer's Guide</i> .	High
rerate	BatchRatingPipeline	Specifies whether the batch-rating pipeline is active during rerating. See "Specifying Whether the Batch Rating Pipeline Is Enabled" in <i>BRM Setting Up Pricing and Rating</i> .	Enabled
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

Tuning Billing Performance by Using Configuration Files

To tune billing performance, you edit entries in the Performance Entries section of the configuration file that is used by all billing utilities (*BRM_home/apps/pin_billd/pin.conf*). You can set values for the following:

- In case of certain BRM search operations that return large amounts of data, you can filter and find accounts in resource sharing groups. See ["Filtering Search Results"](#).
- The number of child threads (or processes) that the billing utilities use. See ["Tuning the Number of Children for Billing Utilities"](#).
- The number of accounts to cache in system memory before BRM processes the accounts. For example, if you specify **50000**, BRM finds and caches 50,000 accounts before processing them. See ["Tuning the Account Cache Size for Billing Utilities \(fetch_size\)"](#).
- How many accounts the child threads can find and move into the cache simultaneously. This ensures that the DM memory is not overloaded. See ["Tuning the Batch Size for Billing Utilities \(per_batch\)"](#).
- The number of accounts sent in batches to the credit card processing service. See ["Tuning the Batch Size for the pin_collect Utility"](#).
- The number of retries in case a deadlock occurs. See ["Specifying the Number of Retries in a Deadlock"](#).
- How accounts cached in system memory are distributed to child threads. See ["Rearranging Accounts to Improve Billing Performance"](#).

In the configuration file, entries preceded by **-pin_billd** apply to all billing utilities. Entries preceded by a specific utility name override the general entry for that utility. In this example, the **children** and **fetch_size** entries are used by all billing utilities, but

the **pin_bill_accts** and **pin_collect** utilities have different values for the **per_batch** entry.

```
- pin_bill children 5
- pin_bill fetch_size 10000
- pin_bill per_batch 2500
- pin_bill_accts per_batch 1000
- pin_collect per_batch 200
```

The **children**, **fetch_size**, and **per_batch** entries are standard configuration options for multithreaded applications. For information about these entries, see "Configuring Your Multithreaded Application" in *BRM Developer's Guide*.

See the comments in the configuration file (*BRM_home/apps/pin_bill/pin.conf*) for more information.

For billing performance issues, see ["Tuning Memory for Billing"](#). For billing performance issues not related to the configuration file, see ["Additional Issues Related to Billing Performance"](#).

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 resource sharing groups:

- **group_members_fetch**: Use this entry to search for members of the group sharing object when the parent group contains many members. The parameter is specified as **- fm_bill group_members_fetch *n***, where *n* is the size of pool (memory in bytes) for the step search. The default value is 0.
- **group_children_fetch**: Use this entry to search for child accounts in groups when the parent group contains many members. The parameter is specified as **- fm_bill group_children_fetch *n***, where *n* is the size of pool (memory in bytes) for the step search. The default value is 0.
- **item_fetch_size**: Use this entry when searching for items. The parameter is specified as **- 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.

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.

3. Save and close the file.

Tuning the Number of Children for Billing Utilities

By default, a billing utility uses five child threads to process accounts. You can increase the number in the **children** entry 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/pin_bill/pin.conf*).
2. In the Performance Entries section, edit the **children** entry:


```
- pin_bill children 5
```
3. Save and close the file.

Tuning the Account Cache Size for Billing Utilities (**fetch_size**)

If there is enough memory, make the value in the **fetch_size** entry equal to or greater than the number of accounts to be billed. If memory is limited, you can use a **fetch_size** entry of 10,000 and a **per_batch** entry of 2500.

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/pin.conf*).
2. In the Performance Entries section, edit the **fetch_size** entry:


```
- pin_bill fetch_size 10000
```
3. Save and close the file.

Tuning the Batch Size for Billing Utilities (**per_batch**)

When processing a batch of accounts before moving them into the cache, the billing utility waits for all children to finish processing before getting another batch of accounts. Therefore, a single account that requires a long processing time can reduce performance because the idle child threads must wait for one child thread to finish processing the account. To minimize this loss in productivity, you can increase the batch size. If you increase the batch size too much, you will run out of memory.

To tune the batch size:

1. Open the billing utilities configuration file (*BRM_home/apps/pin_bill/pin.conf*).
2. In the Performance Entries section, edit the **per_batch** entry:


```
- pin_bill per_batch 2500
```
3. Save and close the file.

Tuning the Batch Size for the **pin_collect** Utility

The **pin_collect** utility uses the **per_batch** entry differently than other billing utilities use it:

- For most billing utilities, the **per_batch** entry specifies the number of accounts that BRM finds and caches in system memory.
- For the **pin_collect** utility, the **per_batch** entry specifies the number of accounts sent in batches to the credit card processing service.

For best performance, the value in the **per_batch** entry for the **pin_collect** utility should be much smaller than the **per_batch** entry for the rest of the billing utilities. Use a value that is equal to the **per_batch** entry that applies to the **pin_bill_accts** entry divided by the value of the **children** entry. (The **pin_bill_accts** value can be set either by a generic **-pin_billd** entry in the billing utility configuration file or by a specific **-pin_bill_accts** entry.)

For example, if the **per_batch** entry for **pin_bill_accts** is 3000, and you use 10 child threads, the **per_batch** entry for the **pin_collect** utility should be 300.

To tune the batch size for the **pin_collect** utility:

1. Open the billing utility configuration file (*BRM_home/apps/pin_bill/pin.conf*).
2. In the Performance Entries section, add a **per_batch** entry for **pin_collect**:

```
- pin_collect per_batch 500
```
3. Save and close the file.

Specifying the Number of Retries in a Deadlock

A deadlock occurs when two or more database sessions have some data locked, and each database session requests lock on the data that the other session has already locked. The database sessions wait on each other to release the lock on the data. Billing utilities, such as **pin_bill_accts** and **pin_inv_accts**, retry the operation in a deadlock. By default, the number of retries is set to 20, but you can change that number. Retries are attempted with no time delay in between. You cannot configure time delay between retries. For information on the deadlock error, see ["Reference Guide to BRM Error Codes"](#).

To specify the number of times to retry an operation when a deadlock occurs:

1. Open the billing utilities configuration file (*BRM_home/apps/pin_bill/pin.conf*).
2. *Uncomment* the **- pin_bill_accts deadlock_retry_count** entry.
3. Edit the **deadlock_retry_count** entry:

```
- pin_bill_accts deadlock_retry_count 20
```
4. Save and close the file.

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 before 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 database 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 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_bill/pin.conf*).
2. In the Performance Entries section, edit the **delta_step** entry:

```
- pin_bill delta_step 10
```
3. Save and close the file.

Improving Performance in Retrieving Purchased Offerings for a Bill Unit

You can improve billing performance while retrieving purchased products and discounts for a bill unit (**/billinfo** object) from the database, by specifying the batch size of the number of services to search at a time in the **MaxServicesToSearch** parameter in the **/config/business_params** object for subscription.

To specify the batch size of the number of services to search:

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

```
pin_bus_params -r -c "Subscription" bus_params_subscription.xml
```

This command creates the XML file **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. Open the **bus_params_subscription.xml.out** file in a text editor.
3. In the **BusParamsSubscription** section, specify the batch size of the number of services to search by changing the value of the **MaxServicesToSearch** tag as follows:

```
<BusParamsSubscription>
...
  <MaxServicesToSearch>5</MaxServicesToSearch>
</BusParamsSubscription>
```

Note: The default value is **5**. The minimum allowed value is **1**. The maximum allowed value depends on the length of the POIDs involved in the search and the search criteria. Depending on the length of the POIDs involved in the search, if you increase the value of **MaxServicesToSearch** and the length of the select statement is more than **2048** characters, the database select statement could fail.

4. Save the file as **bus_params_subscription.xml**.
5. Go to the *BRM_home/sys/data/config* directory and load the change into the **/config/business_params** object by using the following command:

```
pin_bus_params bus_params_subscription.xml
```

Note: To run the utility from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

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

See "Reading an Object and Fields" in *BRM Developer's Guide*.

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

See ["Starting and Stopping the BRM System"](#).

8. Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

For more information, see ["pin_multidb"](#).

Tuning Memory for Billing

If you change the **children** and **per_batch** entries for billing utilities, you should also review the **dm_shmsize** configuration entry for the DM. You can estimate the amount of shared memory required by using the following formulas:

- **To run billing (pin_bill_accts):** The amount of shared memory required, in kilobytes, is equal to 0.45 multiplied by the number specified in the **per_batch** entry.
- **To run credit-card collection (pin_collect):** The amount of shared memory required, in kilobytes, is equal to the number specified in the **children** entry multiplied by the number specified in the **per_batch** entry.

Additional Issues Related to Billing Performance

- To reduce system load, you can split large billing runs into smaller billing runs. See "Reducing Billing Run Loads" 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 "Maintaining Transmission Logs for Billing Transactions" in *BRM Configuring and Running Billing*.
- 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 "Editing the 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.

Tuning Billing Performance by Using Business Parameters

You can improve the billing performance by doing the following:

- [Improving Performance by Skipping Previous Total Unpaid Bill for Open Item Accounting Type](#)
- [Improving Trial Billing Performance by Enabling General Ledger Collection](#)
- [Excluding Searches on Closed Offerings](#)
- [Excluding Searches on Overridden Products](#)

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 by setting the **PerfAdvancedTuningSettings** performance tuning business parameter. Using this business parameter, you can reduce the time taken for calculating the current bill for the open item accounting type.

To skip previous total unpaid bill for open item accounting type:

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 the following line:

```
<PerfAdvancedTuningSettings>0</PerfAdvancedTuningSettings>
```

4. Add **8** to the existing value to skip the previous unpaid bill. For example,

```
<PerfAdvancedTuningSettings>8</PerfAdvancedTuningSettings>
```

For more information on the parameter values, see the **PerfAdvancedTuningSettings** parameter description in [Table 45-3, "Billing business_params Entries"](#).

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.

Note: Setting the **PerfAdvancedTuningSettings** business parameter will impact all the bill units associated with the open item accounting type. To impact specific bill units of an open item accounting type, see ["Improving Performance by Skipping Previous Total for Open Item Accounting Type When Calculating the Current Bill"](#).

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 pathToWorkingDirectory/bus_params_billing.xml
```

where *pathToWorkingDirectory* is the directory in which **bus_params_billing.xml** resides.

Note: To execute it from a different directory, see the description for **pin_bus_params** in *BRM Developer's Guide*.

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

For more information on reading objects by using the Object Browser, see *BRM Managing Customers*. For instructions on using the **testnap** utility, see *BRM Developer's Guide*.
9. Stop and restart the Connection Manager. See ["Starting and Stopping the BRM System"](#).
10. (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 45-3, "Billing business_params Entries"](#).

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 execute it from a different directory, see the description for **pin_bus_params** in *BRM Developer's Guide*.

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

For more information on reading objects by using the Object Browser, see *BRM Managing Customers*. For instructions on using the **testnap** utility, see *BRM Developer's Guide*.
9. Stop and restart the Connection Manager. See "[Starting and Stopping the BRM System](#)".
10. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "[pin_multidb](#)".

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 disabling the **CancelledOfferingsSearch** parameter in the **subscription** instance of the `/config/business_params` object.

You modify the `/config/business_params` object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To disable searches on closed offerings:

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. 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. Locate the **CancelledOfferingsSearch** entry in the **bus_params_subscription.xml.out** file.
3. Set the value of **CancelledOfferingsSearch** to **disabled**, if necessary:
`<CancelledOfferingsSearch>disabled</CancelledOfferingsSearch>`
4. Save this updated file as **bus_params_subscription.xml**.
5. Load the modified XML file into the appropriate **/config/business_params** object in the BRM database.

```
pin_bus_params bus_params_subscription.xml
```

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

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

 For more information on reading objects by using the Object Browser, see *BRM Managing Customers*. For instructions on using the **testnap** utility, see *BRM Developer's Guide*.
7. Stop and restart the Connection Manager. See ["Starting and Stopping the BRM System"](#).
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Excluding Searches on Overridden Products

By default, BRM retrieves the overridden products during the billing process.

If you are not using tailor-made products, you can configure BRM to ignore and not retrieve the overridden products by disabling the **TailormadeProductsSearch** parameter in the **subscription** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To disable searches on overridden products:

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. 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. Locate the **TailormadeProductsSearch** entry in the **bus_params_subscription.xml.out** file.
3. Set the value of **TailormadeProductsSearch** to **disabled**, if necessary:
`<TailormadeProductsSearch>disabled</TailormadeProductsSearch>`
4. Save this updated file as **bus_params_subscription.xml**.
5. Load the modified XML file into the appropriate **/config/business_params** object in the BRM database.

pin_bus_params bus_params_subscription.xml

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

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

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

7. Stop and restart the Connection Manager. See ["Starting and Stopping the BRM System"](#).
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Tuning Billing Performance by Using Business Profiles

To tune billing performance, you set entries in the business profile associated with the bill unit.

For general information about business profiles, see "Managing Business Profiles" in *BRM Managing Customers*.

Improving Performance by Skipping Previous Total for Open Item Accounting Type When Calculating the Current Bill

By default, the previous unpaid bill is calculated even for an open item accounting type 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 the previous bill items.

You can skip the previous total unpaid bill for the open item accounting type when calculating the current bill by setting the **Skip_Prev_Total** business profile key to improve the BRM performance. For making 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 in the *BRM_home/sys/data/config* directory.

2. Set the following business profile key value to **Yes**:

```
<NameValue key="Skip_Prev_Total" value="Yes"/>
```

When the value is set to **Yes**, previous unpaid bill amount *is not* calculated for the current bill.

When the value is set to **No**, previous unpaid bill *is* calculated for 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 "Using testnap" in *BRM Developer's Guide*.

For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

6. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).

For general information about business profiles, see "Managing Business Profiles" in *BRM Managing Customers*.

Improving Performance by Using Multiple Item Configurations

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 the 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 in the *BRM_home/sys/data/config* directory.

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:

```
<NameValuePair 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*.

Caution: Modifying the business profile affects all the 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 "Using testnap" in *BRM Developer's Guide*.

For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
7. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).

For more information about business profiles, see "Managing Business Profiles" in *BRM Managing Customers*.

Improving Performance by Skipping Billing-Time Tax Calculation

BRM can calculate taxes during real-time rating, pipeline batch rating, or billing. See "Choosing When to Calculate Taxes" in *BRM Calculating Taxes*.

Calculating taxes at billing time (deferred taxation) can be a time-consuming process that involves searching all events and items in a billing cycle. It is triggered by regular billing, Bill Now, Bill on Demand, and trial billing.

In some cases, such as for prepaid accounts, taxes typically 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.

Important:

- 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 in the *BRM_home/sys/data/config* directory.
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" />
```

When the value is set to **Yes**, taxes *are not* calculated for the bill unit at billing time.

When the value is set to **No**, taxes *are* calculated for the bill unit 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 "Using testnap" in *BRM Developer's Guide*.
 For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
6. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).

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 the 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
```

See the comments in the configuration file (*BRM_home/apps/pin_inv/pin.conf*) for more information.

Setting the Number of Children for Invoice Utilities

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 must 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)

If enough memory is available, set the value of the **fetch_size** entry to the number of accounts that must 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:
 - `pin_mta fetch_size 4000`
3. Save and close the file.

Setting the Batch Size for Invoice Utilities (per_step)

The recommended value for the **per_step** entry is 100 times the number specified in the **children** entry. Too high a value for the **per_step** entry can overload the DM memory.

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:
 - `pin_mta per_step 250000`
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 14–5 lists the bit values and their significance:

Table 14–5 Invoicing Performance Bit Values

Value	Description
0x00000001	Selects all event types. By default, only events that are configured in the <code>/config/invoice_events</code> object are selected.

Table 14–5 (Cont.) Invoicing Performance Bit Values

Value	Description
0x00000002	<p>Selects only the following event types:</p> <ul style="list-style-type: none"> ■ /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	<ul style="list-style-type: none"> ■ /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

Table 14–5 (Cont.) Invoicing Performance Bit Values

Value	Description
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.
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.

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**.

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

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

7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser or by using the **robj** command with the **testnap** utility.

For information on using **testnap**, see "Using testnap" in *BRM Developer's Guide*.

For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

8. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Improving Data Processing 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.

Note: Statement-handle caching is available with Oracle databases only.

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 Database—pares the statement, executes it, and fetches the results.

Oracle database 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 database parameters must 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 *ratable events* (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 price list. This is enforced by Pricing Center and the Price List Facilities Module (FM) opcodes.

However, many nonratable events may not need to be recorded. For example, event objects that are recorded during account registration, deal purchase, and product 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*.
-

Ratable events that are mapped in the **pin_event_map** file should not be added to the **pin_event_record_map** file. If you specify a ratable event type, the event record map is ignored when the ratable event occurs.

Caution: 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.

Tip: You can also use the Oracle tracing facility to find missing indexes.

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 Customer Center, have little impact on system performance.
- Cycle events are rated faster than usage events.
- Performance decreases when accounts own a large number of products.
- 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.

Tuning Multithreaded Workloads

Usually, 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.

CM and DM RAM and Swap Guidelines

Each CM system should have at least 128 MB of RAM. Too much more than 256 MB might not add additional performance unless there are a large number of connections to the CM.

DM RAM can be lower than 512 MB for smaller installations. For larger installations with eight or more DM CPUs on a single SMP box, 512 MB to 1 GB is recommended. Heavy usage of certain business policies and large searches can greatly increase CM memory requirements.

A typical requirement for swap on a CM/DM system is two to three times the amount of RAM, but this depends upon the operating system and the number of processes running on the CM/DM system.

Hardware Guidelines

- Use large disk controller RAM size.
- Maximize the processor cache size on the database and DM servers. For best performance, use caches that are at least 1 MB.
- Usually, performance is best when there are approximately one and a half times as many CPUs on the CM/DM systems as on the database server system.
- The total number of CM CPUs (regardless of the number of CMs) is approximately 25% - 33% of the total number of CPUs on the database server.
- CM and DM systems need less RAM than the database system.
- CM and DM systems have very low disk requirements. Disks are needed only for the operating system, BRM software, and swap space. Two 9-GB disks on each CM

or DM system is usually enough, as long as these systems are not being used for non-BRM workloads.

Two disks are recommended so that access to temp space, the operating system, swap, and BRM code are not bottlenecked on a single disk.

Improving Network Performance

Any kind of network connection that supports TCP/IP supports BRM (for example, local area network, virtual private network, and PPP).

The network bandwidth needs are relatively simple. BRM has an OLTP footprint (as opposed to Decision Support). Hence, transactions are normally small and network traffic will consist of smaller packets. The only requirement for network connectivity is TCP/IP. The real constraint is to have enough bandwidth between the DM systems and the database server; otherwise, the network might become a bottleneck even under moderate loads.

- Use 100BaseT or FDDI between each DM system and the database server. 10 Mbit Ethernet have too little bandwidth to handle heavy DM-to-database-server traffic. To minimize any collisions, the DM-to-database connections should use a separate physical connection (or a switch) and each of these should be a separate network.
- For switch-connected systems, connect all systems in the BRM configuration by using a single switch. For the largest configurations, use gigabit connections.
- When the test systems have multiple network cards, verify that the operating system network routing tables are configured to avoid bottlenecks. By default (depending on the system), output from the database server system may go through one LAN card, even though you have several configured, and even though input comes in through the multiple cards from different DMs. Examine and fix the routing tables as necessary. Ensure that each DM-to-database-server connection is explicit so all traffic between the two machines goes through the single dedicated physical path. The most common environment where you might find this problem is when there are multiple paths between the same sets of systems.
- For best performance, ensure all the systems in the testing environment are connected to the same hub with no intermediate hops.

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 appear 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.
 - RADIUS Manager with a low number of threads.
 - 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 HP-UX IA64, AIX, Linux, and Oracle Solaris systems, use **vmstat**, **netstat**, and **sar**.
 - HP-UX IA64 has an additional tool called **Glance**.
 - 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 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
```

About Benchmarking

To determine the best possible performance for your system, you must 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 except for the amount of disk space used. If you run the system for a long time, indexes might become imbalanced and must 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 and Maintaining 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:
 - Oracle Solaris: **/etc/system**
 - HP-UX IA64: **/stand/system**
 - Linux: **/etc/sysctl.conf**
 - AIX: Enter the following command to get a list of all parameters:

```
lsattr -E -l sys0
```
- Administrative tools for managing systems:
 - HP-UX IA64: **sam**
 - Oracle Solaris: **Admintool**
 - Linux: **Webmin** and **Easilix**

- AIX: **smit**
- The **pin.conf** files for CMs, DMs, and clients such as RADIUS Manager.
- For Oracle database, 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 Database 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?
 - CMMF
 - CM Proxy
 - CM
 - DM
 - Optional managers such as RADIUS Manager
 - 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 products per account?
- What is the largest quantity of products 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 Database Configuration

- What is the Oracle database version?
- How is the **init.ora** file configured?

The following **init.ora** Oracle database 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?

Optimizing Pipeline Manager Performance

This chapter describes tools and techniques you can use to optimize Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager performance.

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

- [Configuring Pipeline Manager](#)
- "About Pipeline Rating" in *BRM Configuring Pipeline Rating and Discounting*

Pipeline Manager Optimization Overview

When you optimize Pipeline Manager performance, your objective is to increase the percentage of CPU time spent on user processes and to decrease the percentage of time spent idle or on system processes.

Complete performance tuning requires much testing. Due to the complexity of most Pipeline Manager configurations, optimization is a highly iterative process. You cannot configure options formulaically, but you must test many configurations and then implement the optimal configuration. This chapter describes optimization methods to guide your testing for a given set of hardware resources.

Software optimization techniques can include modifying the following:

- The number and type of function modules.
- The design of custom iScripts and iRules.
- The number of system threads used by a pipeline.
- The number of call data record (CDR) files configured for a transaction.
- The number of pipelines configured for the system.

Note: Available hardware resources can constrain the usefulness of some optimization techniques. For example, if your system has only a few CPUs, you probably will not see performance gains by using multithreaded mode.

Key Metrics for Measuring Performance

When evaluating performance improvement, the primary metrics to monitor are:

- The ratio of CPU time spent on system processes to CPU time spent on user processes. This ratio should be about 1 to 2 or lower.

- The percentage of idle CPU time. This percentage should be 20 percent or less.
- The results of performance tests using sample CDR files.

About Measuring Pipeline Manager Performance

You use the Pipeline Manager instrumentation feature as the primary tool for measuring Pipeline Manager performance. See ["Measuring System Latencies with Instrumentation"](#) for more information. When instrumentation is enabled, information about how much time in microseconds each function module uses to process a certain number of files is written to the pipeline log file (**pipeline.log**). You then use this information when you apply some optimization techniques.

Note: For more information on the pipeline log, see "LOG" in *BRM Configuring Pipeline Rating and Discounting*.

Other Pipeline Manager performance monitoring tools are:

- Monitor event data record (EDR) throughput. See ["Monitoring Pipeline Manager EDR Throughput"](#).
- Monitor recent log files. See ["Getting Recent Pipeline Log File Entries"](#).
- Monitor memory usage. See "Memory Monitor" in *BRM Configuring Pipeline Rating and Discounting*.

Information Requirements

Before you optimize Pipeline Manager, be familiar with your existing system configuration, such as:

- Total system memory.
- Other (nonpipeline) processes running on the Pipeline Manager system that will share system resources.
- The number and types of pipelines required for your business logic or planned load balancing.
- The expected load for each pipeline.
- Whether your business logic is more CPU intensive or I/O intensive. (For example, if you use the FCT_Discount module, your business logic is likely to be more CPU intensive.)

Testing Requirements

Before you optimize Pipeline Manager, you should have a set of error-free sample CDRs that resemble those used in your production system.

Optimizing Pipeline Manager

To optimize Pipeline Manager, consider the following actions:

- (Oracle Solaris, Linux, HP-UX IA64, and AIX) Be sure that OS-specific system configurations were put in place during installation. See the following topics in *BRM Installation Guide*:
 - Creating a User and Configuring Environment Variables

- (Solaris) Setting Maximum Open Files on Solaris
- (Solaris) Configuring Memory Allocation and Block Transfer Mode on Solaris Systems
- (Linux) Setting Maximum Open Files on Linux
- (HP-UX IA64) Setting Maximum Open Files on HP-UX IA64

Important: For HP-UX IA64, you must set the `_M_ARENA_OPTS` and `_M_CACHE_OPTS` environment variables to achieve acceptable system performance.

- (AIX) Setting Maximum Open Files on AIX
- Configure pipelines to run in either single-threaded or multithreaded mode. See ["Configuring Single-Threaded or Multithreaded Operation"](#) for more information.
It is especially important to maximize the performance of the `DAT_AccountBatch` and `DAT_BalanceBatch` modules. See:
 - [Configuring the DAT_AccountBatch Module Database Connections and Threads](#)
 - [Configuring Threads for DAT_BalanceBatch Connections](#)
- Configure function pools within pipelines. See ["Optimizing a Pipeline by Using Function Pools"](#) for more information.
- If you have CDR files smaller than a few thousand records, consider grouping multiple CDR files into one transaction. See ["Combining Multiple CDR Files into One Transaction"](#) for more information.
- Configure multithreading in the Output Controller. See ["Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams"](#) for more information.
- Add additional pipelines. See ["Configuring Multiple Pipelines"](#) for more information.
- Verify that any custom iScripts and iRules are efficiently designed. See ["Optimizing Function Modules"](#) for more information.
- Configure event and service mapping to only supply the Pipeline Rating Engine with the services being rated. See ["Mapping Events and Services"](#) in *BRM Setting Up Pricing and Rating*.
- Configure the `DAT_USC_Map` module to improve startup performance. See ["Configuring the DAT_USC_Map Module for Startup Performance"](#) and ["DAT_USC_Map"](#) in *BRM Configuring Pipeline Rating and Discounting*.

Troubleshooting Pipeline Performance

Use the following checklist to troubleshoot drops in performance.

- If you installed a patch, find out if the patch changed operating system functions, such as threading or memory management, or made any changes to Pipeline Manager framework modules.
- Check recent customizations, such as iScripts. Look for customizations that might impact database access or hash usage.

- Use database monitoring tools to monitor the Pipeline Manager database to see if there is a lot of activity. If so, check which queries are used and which indexes are used. This might point to the data involved, which might point to the module processing that data.
- Use a monitoring command such as **iostat** to check I/O activity.
- Use a memory monitoring command such as **prstat**, **vmstat**, or **sar** to check if the Pipeline Manager memory usage has changed. If Pipeline Manager uses an unexpected amount of memory, check for duplicate keys related to buffers and call assembly.
- Check for large numbers of files in the following directories:
 - **in**
 - **err**
 - **done**
 - **dupl**
 - **assembl**
 - **rej**Delete old files that are no longer needed.
- Look for bottlenecks in the pipeline by using the **prstat** command and the thread ID in the **process.log** file to identify slow threads. Check for:
 - **icx** (involuntary context switch)
 - **vcx** (voluntary context switch)
 - **scl** (system call)
 - **slp** (sleep)
- Check the **pipeline.log** file for records of a large amount of rollbacks.

Optimizing Function Modules

Slow function modules can be very detrimental to overall Pipeline Manager performance. To optimize individual function modules:

1. Identify the high latency modules by using instrumentation. See "[Measuring System Latencies with Instrumentation](#)" for more information.
2. Check if the high latency modules can be optimized. For example, you might discover that the business logic used in high latency iScripts or iRules can be redesigned to improve performance.

Configuring Single-Threaded or Multithreaded Operation

You configure pipelines to run in single-threaded or multithreaded mode by using the **MultiThreaded** registry entry in the registry file.

- **Single-threaded mode:** Use this mode if you are using a system with just a few CPUs and limited RAM.

In a single-threaded environment, pipelines use a single thread to run all modules and only one CPU is used for each pipeline.

If the **MultiThreaded** registry entry is not included in the registry file, pipelines will by default run in multithreaded mode.

Note: Business logic can prevent the setup of multiple pipelines.

- **Multithreaded mode:** Use this mode if your system has many CPUs.

In a multithreaded environment, pipelines use three or more threads to process each transaction. By default, one thread is used for the input module and one for the output module. An additional thread is used for each function pool that you configure to process function modules.

For information on optimizing pipelines when using multithreaded mode, see:

- [Assigning Multiple Threads to Process Function Modules](#)
- [Optimizing a Pipeline by Using Function Pools](#)

For information about the **MultiThreaded** registry entry, see "Pipeline Controller" in *BRM Configuring Pipeline Rating and Discounting*.

To configure single-threaded or multithreaded operation:

1. Open the registry file in a text editor.
2. Set the input controller **MultiThreaded** registry entry to the appropriate value:
 - **True** to configure the pipeline for multithreaded processing.
 - **False** to configure the pipeline for single-threaded processing.

```
Pipelines
{
    PipelineName
    {
        MultiThreaded = value
        ...
    }
}
```

3. Restart the pipeline. See ["Starting and Stopping Individual Pipelines"](#) for more information.

Note: For more information on all registry entries pertaining to individual pipelines, see "Pipeline Controller" in *BRM Configuring Pipeline Rating and Discounting*.

Reducing Startup Times with Parallel Loading

You can reduce your startup times by configuring Pipeline Manager to:

- Load all pipelines in parallel.
- Load data modules in parallel.
- Load function modules in parallel.

By default, Pipeline Manager loads pipelines, data modules, and function modules sequentially.

To enable parallel loading, use the Parallel Load Manager module:

1. Open the registry file in a text editor.
2. Configure the **ifw.ParallelLoadManager** section of the registry file:
 - Set the **Active** registry entry to **True**.
 - Set the **NumberOfThreads** registry entry to the number of threads you want Pipeline Manager to use for loading your pipelines, data modules, and function modules.

For example:

```
ifw
{
    ...
    ParallelLoadManager
    {
        Active = True
        NumberOfThreads = 4
    }
    ...
}
```

3. Restart the pipeline. See ["Starting and Stopping Individual Pipelines"](#) for more information.

Assigning Multiple Threads to Process Function Modules

If a pipeline is configured for multithreaded processing and you have idle CPU resources, you might be able to increase performance by grouping function modules into two or more function pools. The pipeline runs each function pool in a separate thread.

Important: Adding too many function pools to a pipeline can decrease performance because the buffers between the threads consume system CPU overhead and RAM. (Typically, two to six function pools is optimal.)

Tip: If you are using a high-latency module such as FCT_AccountBatch or FCT_Discount and have sufficient hardware resources, assign the module to its own function pool and test for performance improvement.

To create a separate thread for an individual function module or a group of function modules, you use the **FunctionPool** registry entry.

Important: Before you perform this procedure, read ["Optimizing a Pipeline by Using Function Pools"](#) for more information.

1. Submit some sample CDRs to the pipeline with instrumentation enabled. See ["Measuring System Latencies with Instrumentation"](#) for more information.
2. Locate the instrumentation results in the **pipeline.log** file.
3. Open the registry file in a text editor.

4. Using the instrumentation data, reduce the processing time required by the slowest function pool by:
 - (Optional) Adding an additional function pool to the **Functions** section of the registry file.
 - Shifting one or more modules from a function pool to an adjacent function pool.

The objective is to make the processing times of all function pools as similar as possible.
5. Save the registry file.
6. Restart the pipeline. See ["Starting and Stopping Individual Pipelines"](#) for more information.
7. Measure pipeline performance with the sample CDRs by measuring transaction start times and end times in the **pipeline.log** file.
8. Go to Step 3 and repeat testing until optimal results are achieved.

Configuring the DAT_AccountBatch Module Database Connections and Threads

To improve performance, you can configure multiple DAT_AccountBatch connections to the BRM database. See "DAT_AccountBatch" in *BRM Configuring Pipeline Rating and Discounting*. Configure the following registry entries:

- Use the **Threads** registry entry to specify the number of threads. Set this value to at least the number of CPUs in the system. Increasing the number of threads increases performance, up to a point. Specifying too many threads decreases performance.
The default is 4.
- Use the **Connections** registry entry to specify the number of connections to the database. This value must be at least one more than the number of threads.
The default is 5.
- Use the **LoadPercentage** registry entry to specify the percentage of account POIDs to store locally when determining the account blocks for which each thread is responsible.
Values must be greater than 0.000000 and less than or equal to 100.0.
The default is 10.

Setting the Hash Map Size for Threads

You can use the following DAT_AccountBatch registry entries to set the temporary hash map size built for each thread. Each entry controls the hash map size for a different type of data; for example, accounts, logins, and services.

In general, larger maps perform better but consume more memory. Smaller maps save memory but can slow down Pipeline Manager startup. Very low numbers can dramatically slow down Pipeline Manager startup.

The default system-calculated value uses the following formula:

$((\text{number of accounts} / \text{number of threads}) * 2).$

The registry entries are:

- **ThreadAccountHashMapSize**: Used for account data.

Important: Changing the default system-calculated values for this entry is not recommended. Replacing this entry with one larger than the default wastes memory. Replacing this entry with one smaller than the default slows Pipeline Manager startup.

- **ThreadGroupSharingChargesHashMapSize:** Used for charge sharing group data. The system-calculated default value might not be appropriate.

If your accounts average fewer than two or more than four GroupSharingCharges per account, use the following formula as a guideline to calculate an entry:

$$(((\text{number of accounts} * \text{average number of GroupSharingCharges per account}) / \text{number of threads}) * 75\%).$$

- **ThreadGroupSharingDiscountsHashMapSize:** Used for discount sharing group data. The system-calculated default value might not be appropriate.

If your accounts average fewer than two or more than four GroupSharingDiscounts per account, use the following formula as a guideline to calculate an entry:

$$(((\text{number of accounts} * \text{average number of GroupSharingDiscounts per account}) / \text{number of threads}) * 75\%).$$

- **ThreadGroupSharingProfilesHashMapSize:** Used for profile sharing group data. The system-calculated default value might not be appropriate.

If your accounts average fewer than two or more than four profile sharing groups per account, use the following formula as a guideline to calculate an entry:

$$(((\text{number of accounts} * \text{average number of GroupSharingProfiles per account}) / \text{number of threads}) * 75\%).$$

- **ThreadLoginHashMapSize:** Used for login data. The system-calculated default value is appropriate for most implementations.

If your accounts average more than four logins per account, use the following formula as a guideline to calculate an entry:

$$(((\text{number of accounts} * \text{average number of logins per account}) / \text{number of threads}) * 75\%).$$

- **ThreadServiceHashMapSize:** Used for service data. The system-calculated default value is appropriate for most implementations.

If your accounts average more than four services per account, use the following formula as a guideline to calculate an entry:

$$(((\text{number of accounts} * \text{average number of services per account}) / \text{number of threads}) * 75\%).$$

Locking Objects during DAT_AccountBatch Processing

You can set the number of pre-allocated mutex objects that are used to lock individual objects during processing to prevent multiple threads from contending for access to the same object. You can use different settings for account, login, and service objects by setting the following DAT_AccountBatch registry entries:

- **AccountLocks**
- **LoginLocks**
- **ServiceLocks**

Usually, the default value for these entries should be appropriate. If you use a larger value, less allocation is needed for additional mutex objects during processing, but more memory is used.

The default for all entries is 10.

Configuring Threads for DAT_BalanceBatch Connections

Use the following DAT_BalanceBatch registry entry to configure connections to the BRM database:

- **Threads:** Specifies the number of threads for loading the balance data from the BRM database. The number of threads must be smaller than or equal to the number of connections.

The default is 4.

- **ThreadHashMapSize:** Specifies the size of the hash map in each thread used for loading balance data from the BRM database.

The default is 1024.

Improving Pipeline Manager Startup Performance

For information about improving Pipeline Manager startup performance, see:

- [Improving DAT_BalanceBatch Loading Performance](#)
- [Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing](#)

Improving DAT_BalanceBatch Loading Performance

The DAT_BalanceBatch module uses the noncurrency resource validity to select the noncurrency subbalances to load from the BRM database into pipeline memory. If the noncurrency resource validity is not configured, at Pipeline Manager startup, DAT_BalanceBatch selects the subbalances that were valid for 366 days by default. When the BRM database contains a large number of noncurrency subbalances, loading them leads to increased Pipeline Manager startup times.

To improve Pipeline Manager startup performance, you can set the noncurrency resource validity to specify the subbalances to load. See "Specifying Which Non-Currency Subbalances to Load on Startup" in *BRM Setting Up Pricing and Rating* for more information.

Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing

The DAT_AccountBatch and DAT_BalanceBatch modules use multithreaded framework to load account and balance data from the BRM database into Pipeline Manager memory. The modules group the accounts and balances into batches or jobs. Multiple worker threads run in parallel to process the jobs. When a thread completes processing, it is assigned another job from the jobs pool, which improves load balancing between the threads and increases Pipeline Manager startup performance.

By default, the number of jobs per thread is 3, which is appropriate in most installations to achieve load balancing. However, if thread loading times vary greatly, you can use the **PerThreadJobsCount** entry in the DAT_AccountBatch registry and the **BalancesPerThreadJobsCount** entry in the DAT_BalanceBatch registry to adjust the number of jobs per thread.

Important: Setting the number of jobs per thread to a large number can outweigh the performance gain because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). To adjust the number of accounts or balances per job, you can do this by increasing or decreasing the number of threads. However, when the number of accounts or balances is too small, the data modules use one thread to optimize performance.

Breaking Up Large Nested Subsections in Registry Files

Pipeline Manager can encounter parser stack overflow errors when a pipeline registry section contains a large number of nested subsections.

You can break up large nested subsections and prevent parser stack overflow errors by using anonymous blocks in your registry file. An anonymous block consists of a nested subsection with braces { } and no subsection name, as shown below:

```
#-----
# Input section
#-----
Input
{
    UnitsPerTransaction = 1

    InputModule
    {
        { # <-- Beginning of Anonymous Block
            ModuleName = INP_GenericStream
            Module
            {
                Grammar = ./formatDesc/Formats/Solution42/SOL42_V670_REL_InGrammar.dsc
                DefaultOutput = TELOutput

                InputStream
                {
                    ModuleName = EXT_InFileManager
                    Module
                    {
                        InputPath = ./data/incollect/reprice/in
                        InputPrefix = test_
                        InputSuffix = .edr
                        ...
                    }
                } # end of InputStream
            } # end of InputModule
        } # --> End of Anonymous Block
    } # end of InputDataPool
}
```

You can place anonymous blocks in any location and at any hierarchy level of the registry file. For the best effect, divide large sections by placing an anonymous block around a group of smaller subsections. This breaks up the section without affecting the hierarchy of the subsections enclosed within the anonymous block.

Optimizing a Pipeline by Using Function Pools

In general, the performance of a multithreaded pipeline varies directly with its slowest thread. The objective of optimizing a multithreaded pipeline is to group the function modules into function pools so that the slowest function pool is as fast as possible. In this environment, faster threads wait a minimum amount of time for data to be delivered or processed by slower threads.

Important: Adding too many function pools to a pipeline can decrease performance because the buffers between the threads consume system CPU overhead and RAM. (Typically, two to six function pools is optimal.)

You use instrumentation results to guide function pool configuration. Instrumentation results indicate how many microseconds are required by each module to process a given number of requests. You use this information to add function pools or regroup the modules in existing function pools.

Tip: You cannot improve performance by adding function pools or shifting function modules to adjacent function pools if your *slowest* function pool:

- Has one function module in it. (or)
- Is faster than the input or output module.

You might be able to improve performance by *reducing* the number of function pools as long as the slowest function pool is *faster* than the output module. (Any performance gain comes from the reduced number of buffers. Fewer buffers require less system process overhead.)

Adding Function Pools

You might improve system performance by adding one or more function pools.

The following example shows a high-level schema of a portion of a registry file for a pipeline called ALL_RATE:

Note: For information on the buffers between the function pools, see ["Configuring Buffers"](#) for more information.

```
input {...}

Functions
{
    PreProcessing
    {
        FunctionPool
        {
            module_1 {}
            module_2 {}
            module_3 {}
        }
    }
}
```

```
Buffer1 {...}

Rating
{
    FunctionPool
    {
        module_4 {}
        module_5 {}
        module_6 {}
    }
}

Buffer2 {...}

PostRating
{
    FunctionPool
    {
        module_7 {}
        module_8 {}
    }
}

output {...}
```

The instrumentation output in the **pipeline.log** file reveals the following latencies for each module for processing a fixed set of test transactions:

Note: For simplicity, the sample latencies have been rounded to the nearest 5,000,000 microseconds.

```
15.03.2004 13:25:07 testserver ifw IFW NORMAL 00516 -
(ifw.Pipelines.ALL_RATE.Functions.PreProcessing)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_1.Module, 40000000
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_2.Module, 15000000
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_3.Module, 45000000

15.03.2004 13:25:07 testserver ifw IFW NORMAL 00516 -
(ifw.Pipelines.ALL_RATE.Functions.Rating)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_4.Module, 65000000
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_5.Module, 30000000
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_6.Module, 90000000

15.03.2004 13:25:07 testserver ifw IFW NORMAL 00516 -
(ifw.Pipelines.ALL_RATE.Functions.PostRating)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.Postrating.FunctionPool.module_7.Module, 35000000
ifw.Pipelines.ALL_RATE.Functions.Postrating.FunctionPool.module_8.Module, 50000000
```

This output is summarized in [Table 15–1](#):

Table 15–1 Example 1 Module Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	100,000,000
module_2	15,000,000	PreProcessing	100,000,000
module_3	45,000,000	PreProcessing	100,000,000
module_4	65,000,000	Rating	185,000,000
module_5	30,000,000	Rating	185,000,000
module_6	90,000,000	Rating	185,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

The total latency in this configuration is 185,000,000; this represents the microseconds used by the slowest function pool.

Figure 15–1 shows that about a third of the CPU cycles used by the function pool threads are idle:

Figure 15–1 Unused CPU Cycles Example 1

PreProcessing function pool	Unused CPU cycles
Rating function pool	
PostRating function pool	Unused CPU cycles

In this example, the pipeline can be optimized if module_6 is assigned to its own function pool, as in this revised sample:

```
input {...}

Functions
{
    PreProcessing
    {
        FunctionPool
        {
            module_1 {}
            module_2 {}
            module_3 {}
        }
    }

    Buffer1 {...}

    Rating
    {
        FunctionPool
        {
            module_4 {}
        }
    }
}
```

```

        module_5 {}
    }

    Buffer2 {...}

    Discounting
    {
        functionpool
        {
            module_6 {}
        }
    }

    Buffer3 {...}

    PostRating
    {
        FunctionPool
        {
            module_7 {}
            module_8 {}
        }
    }

    output {...}

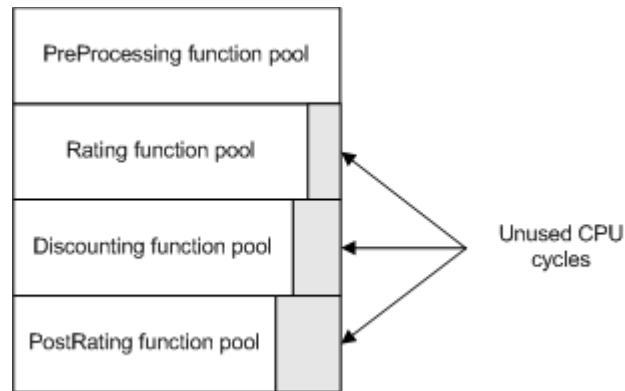
```

The latency table now appears as shown in [Table 15–2](#):

Table 15–2 Example 2 Modules Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	100,000,000
module_2	15,000,000	PreProcessing	100,000,000
module_3	45,000,000	PreProcessing	100,000,000
module_4	65,000,000	Rating	95,000,000
module_5	30,000,000	Rating	95,000,000
module_6	90,000,000	Discounting	90,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

Total function module latency in the new configuration is 100,000,000 microseconds, equivalent to the latency of the PreProcessing function pool. Less than eight percent of function pool CPU cycles are now idle as shown by the gray cycles in [Figure 15–2](#):

Figure 15–2 Unused CPU Cycles Example 2

Shifting Modules between Function Pools

Adding an additional function pool can decrease performance in some situations (see ["Adding Function Pools"](#) for more information). This can occur if the system overhead for the additional buffer more than offsets the performance gains from a faster highest-latency function pool. When this occurs, you might be able to improve performance by keeping the number of function pools constant and shifting modules to adjoining function pools.

In the sample above, if adding an additional function pool decreased performance, you could return to using three function pools and then move module 4 to the end of the PreProcessing function pool as shown in [Table 15–3](#):

Table 15–3 Example 3 Modules Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	165,000,000
module_2	15,000,000	PreProcessing	165,000,000
module_3	45,000,000	PreProcessing	165,000,000
module_4	65,000,000	PreProcessing	165,000,000
module_5	30,000,000	Rating	120,000,000
module_6	90,000,000	Rating	120,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

Total function module latency in the new configuration is 165,000,000 microseconds. This is equivalent to the latency of the PreProcessing function pool. Though performance gains might be more modest than in the first scenario (where a new function pool was added), the performance gain is more certain because no additional buffer overhead was added.

Configuring Buffers

In a multithreaded pipeline, each pair of consecutive threads communicates through a buffer. Because each function pool is assigned a thread, you must configure a buffer between consecutive function pools.

You configure the buffers between function pool sections in the pipeline registry file. Normally, each buffer can be configured as follows:

```
Buffer1
{
    Size = 100
}
```

Important: On Solaris systems, you should configure block transfer mode. See "Block Transfer Mode" in *BRM Installation Guide*.

Combining Multiple CDR Files into One Transaction

Pipeline Manager is generally more efficient when it processes large CDR files. If a pipeline receives and processes small CDR files, you can improve processing performance by combining multiple CDR input files into one pipeline transaction. You use the **UnitsPerTransaction** registry entry in the input controller to implement this functionality. See "Input Controller" in *BRM Configuring Pipeline Rating and Discounting*.

The **UnitsPerTransaction** entry specifies the number of CDR input files that make up a transaction. By default, each CDR file forms its own transaction.

Note: The optimal transaction size depends on your system configuration and pricing model. In general, most system configurations perform best when the total number of CDRs, which is the average number of CDRs per input file multiplied by the number of input files in the transaction, is greater than 10,000.

If the **UnitsPerTransaction** value is greater than 1, you can use the **SequenceGeneration** registry entry in the output controller to specify whether the pipeline generates one output file per CDR input file or one output file for the entire transaction (see "Output Controller" in *BRM Configuring Pipeline Rating and Discounting*). Pipeline Manager performance is generally faster when one output file is generated for the entire (multi-CDR) transaction.

To combine multiple CDR files into one transaction:

1. In the **Input** section of the registry file, set the **UnitsPerTransaction** entry to the number of CDR input files that make up one transaction. For example, set **UnitsPerTransaction** to 100 to combine 100 CDR input files into one transaction.

Note: The default **UnitsPerTransaction** value is 1.

```
Input
{
    ...
    UnitsPerTransaction = 100
    ...
}
```

2. (Optional) In the **Output** section of the registry file, set the **SequenceGeneration** entry to **Transaction**. This configures the pipeline to generate one output file for the entire transaction.

Note: The default **SequenceGeneration** value is **Units**, which configures the pipeline to generate one output file per CDR input file.

```
Output
{
    ...
    SequenceGeneration = Transaction
    ...
}
```

3. Stop and restart the pipeline. See ["Starting and Stopping Individual Pipelines"](#).

Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams

You can enhance Pipeline Manager throughput by configuring multithreading in the Output Controller. This enables Pipeline Manager to write multiple EDRs in parallel when the EDRs are associated with multiple output streams.

Important: Enable multithreading in the Output Controller only if the EDRs are associated with multiple output streams.

Enabling multithreading may cause an increase in the overall memory usage of the Output Controller. However, the memory usage becomes constant after processing EDRs for some time.

To configure multithreading in the Output Controller:

1. Open the registry file (for example, *Pipeline_home/conf/wireless.reg*) in a text editor.
2. In the **MultiThreading** section, do the following:
 - Set the **Active** registry entry to **True**.
 - Set the **NumberOfThreads** registry entry to the number of threads you want the Output Controller to create for Pipeline Manager to write multiple EDRs in parallel.
 - Set the **BatchSize** registry entry to the appropriate value:
 - **0** indicates that the Output Controller does not run in batch mode.
 - A value greater than **0** indicates that the Output Controller operates in batch mode with the batch size equal to the specified value.

For example:

```
Output
{
    ...
    ...
    MultiThreading
    {
        Active = True
        NumberOfThreads = 5
        BatchSize = 500
    }
}
```

```
}
```

3. Save and close the file.
4. Restart the pipeline. See ["Starting and Stopping Individual Pipelines"](#).

For information about the **MultiThreading** registry entry, see "Output Controller" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Multiple Pipelines

If you have high transaction throughput requirements and additional system resources, you might improve system performance by running multiple pipelines that perform the same function.

In general, consider running multiple pipelines if:

- Your system has a relatively large number of CPUs.
- The order of the input streams is not important.

Note: When you use the FCT_CallAssembling or the FCT_DuplicateCheck module, you must process the EDRs for the same account in the same pipeline. See "Using Duplicate Check with Multiple Pipelines" and "Assembling Calls with Multiple Pipelines" in *BRM Configuring Pipeline Rating and Discounting*.

Tip: If you configure multiple pipelines and your system is running at near full capacity on a limited number of CPUs, test running the pipelines in single-threaded mode. This configuration reduces the buffer memory allocation requirement and thread-handling overhead. To enable single-threaded operation, set the **MultiThreaded** entry to **False**. See ["Assigning Multiple Threads to Process Function Modules"](#) for more information.

Customizing Flists Sent to a Real-Time Pipeline

You can configure the fields included in flists sent to a real-time pipeline by using the **load_pin_rtp_trim_flist** utility. See "load_pin_rtp_trim_flist" in *BRM Developer's Guide*. This utility is useful for:

- Improving system efficiency by removing (trimming) fields that the pipeline does not use.
- Supporting custom iScripts and iRules in the pipeline by adding fields to default flists that are not included in the flists by default.

To optimize the set of fields sent to a real-time pipeline:

1. Determine which fields are required by the real-time pipeline.
2. Create an XML file that describes the fields to be sent to the real-time pipeline based on one or more event types.
3. Load the XML file using the **load_pin_rtp_trim_flist** utility.

Configuration Object Dot Notation

The `load_pin_rtp_trim_flist` utility creates a configuration object (`/config/rtp/trim_flist`). This object is used to create the trimmed flists.

The configuration object uses dot notation. For example, the `PIN_FLD_STATUS_FLAGS` field at the end of this portion of a sample flist:

```
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1   PIN_FLD_POID          POID [0] 0.0.0.1 /account 10243 13
1   PIN_FLD_MOD_T         TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
1   PIN_FLD_ACCOUNT_NO    STR [0] "0.0.0.1-10243"
1   PIN_FLD_CURRENCY      INT [0] 840
1   PIN_FLD_BILL_WHEN     INT [0] 1
1   PIN_FLD_LAST_BILL_T   TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1   PIN_FLD_BAL_GRP_OBJ   POID [0] 0.0.0.1 /balance_group 8323 4
1   PIN_FLD_SERVICE_INFO  SUBSTRUCT [0] allocated 32, used 32
2     PIN_FLD_STATUS      ENUM [0] 10100
2     PIN_FLD_STATUS_FLAGS INT [0] 0
```

is represented as:

```
PIN_FLD_INHERITED_INFO.PIN_FLD_SERVICE_INFO.PIN_FLD_STATUS_FLAGS
```

in the configuration object.

About the *field_list.xml* File

The *field_list.xml* file specifies the fields from the `/account` and `/service` objects that are included in the flist that is sent to Pipeline Manager. You can define conditions in `<EventMap>` sections in the XML file that indicate which fields should be included in the flist depending on the event type.

The following example shows the XML file structure with session and provisioning event filters:

```
<EventMapList>

  <!--* The following event map specifies fields sent
    * when the event type is exactly /event/session -->

  <EventMap>
    <Event>
      <Type>/event/session</Type>
      <Flags>0</Flags>
    </Event>
    <RequiredField>

      <!-- List of fields sent put here. -->

    </RequiredField>
  </EventMap>

  <!--* The following event map specifies fields sent
    * when the event type starts with /event/session/ -->

  <EventMap>
    <Event>
      <Type>/event/session/</Type>
      <Flags>1</Flags>
    </Event>
    <RequiredField>
```

```
<!-- List of fields sent put here. -->

</RequiredField>
</EventMap>

<!--* The following event map specifies fields sent
* when when a provisioning event matches any of three conditions. -->

<EventMap>
  <Event>
    <Type>/event/provisioning</Type>
    <Flags>0</Flags>
  </Event>
  <Event>
    <Type>/event/provisioning/session</Type>
    <Flags>0</Flags>
  </Event>
  <Event>
    <Type>/event/provisioning/</Type>
    <Flags>1</Flags>
  </Event>
</RequiredField>

  <!-- List of fields sent put here. -->

  </RequiredField>
</EventMap>

<!--* The following event map specifies fields sent when none of the
* above conditions are true. -->

<EventMap>
  <Event>
    <Type>*</Type>
    <Flags>1</Flags>
  </Event>
</RequiredField>

  <!-- List of fields sent put here. -->

  </RequiredField>
</EventMap>
</EventMapList>
```

The **Flags** tag in the XML file specifies event matching criteria.

- A **Flags** value of **0** specifies that an exact match is required.
- A **Flags** value of **1** specifies that the event type must start with the string specified in the **Type** tag. The value **1** is also used when indicating **Type** value asterisk (*). This value matches all event types.

Important: Search order is important. The fields included with the flist are the fields specified in the first event map section of the XML file where the event type matches the string in the **Type** field.

You can use the sample XML fields list (*BRM_home/sys/data/config/pin_config_rtp_trim_flist.xml*) as a base for your custom XML file.

For a detailed example using session event filters, see ["Usage Example"](#).

Mapping Events to Flists

Because one flist can be used by more than one event, you can specify the relationship between an event and the flist.

For example, the following section is of an event map XML file:

```
<EventMap>
  <Event>
    <Type>/event/session</Type>
    <Flags>0</Flags>
  </Event>
  <Event>
    <Type>/event/session/</Type>
    <Flags>1</Flags>
  </Event>
```

is mapped to an flist as follows:

```
0 PIN_FLD_EVENT_MAP          ARRAY [0] allocated 20, used 8
1   PIN_FLD_EVENTS           ARRAY [0] allocated 20, used 8
2     PIN_FLD_EVENT_TYPE     STR [0] "/event/session"
2     PIN_FLD_FLAGS          INT [0] 0
1   PIN_FLD_EVENTS           ARRAY [1] allocated 20, used 8
2     PIN_FLD_EVENT_TYPE     STR [0] "/event/session/"
2     PIN_FLD_FLAGS          INT [0] 1
```

Usage Example

An unmodified flist might look like the sample shown in ["Sample Unmodified Flist"](#). However, in this example, Pipeline Manager only requires subsets of fields listed in ["Sample Fields Required by Pipeline Manager"](#) depending on the event type.

In this example, to implement the trimmed flist:

1. Create the XML file shown in ["sample.xml File"](#) to modify the default list of fields (["Sample Unmodified Flist"](#)) included in the flist.
2. Load the XML file using the utility:

```
load_pin_rtp_trim_flist -f sample.xml [-v] [-d]
```

Sample Unmodified Flist

The following is the default (untrimmed) list of fields sent to Pipeline Manager.

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /event/session -1 0
0 PIN_FLD_EVENT              SUBSTRUCT [0] allocated 25, used 25
1   PIN_FLD_POID             POID [0] 0.0.0.1 /event/session -1 0
1   PIN_FLD_NAME             STR [0] "Activity Session Log"
1   PIN_FLD_USERID           POID [0] 0.0.0.1 /service/pcm_client 1 0
1   PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 10243 0
1   PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
1   PIN_FLD_START_T          TSTAMP [0] (1065785673) Fri Oct 10 04:34:33 2003
1   PIN_FLD_END_T            TSTAMP [0] (1065785683) Fri Oct 10 04:34:43 2003
1   PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.1 /service/ip 11907 1
1   PIN_FLD_SYS_DESCR        STR [0] "Session: generic"
1   PIN_FLD_RUM_NAME         STR [0] "Duration"
1   PIN_FLD_UNIT             ENUM [0] 1
1   PIN_FLD_TOD_MODE         ENUM [0] 2
```

```

1  PIN_FLD_NET_QUANTITY DECIMAL [0] 60.000000000000000
1  PIN_FLD_MIN_QUANTITY DECIMAL [0] 60.000000000000000
1  PIN_FLD_INCR_QUANTITY DECIMAL [0] 60.000000000000000
1  PIN_FLD_MIN_UNIT      ENUM [0] 2
1  PIN_FLD_INCR_UNIT     ENUM [0] 2
1  PIN_FLD_ROUNDING_MODE ENUM [0] 1
1  PIN_FLD_TIMEZONE_MODE ENUM [0] 1
1  PIN_FLD_RATED_TIMEZONE_ID STR [0] "GMT-08:00"
1  PIN_FLD_TIMEZONE_ADJ_START_T TSTAMP [0] (1065760473) Thu Oct 09 21:34:33 2003
1  PIN_FLD_TIMEZONE_ADJ_END_T TSTAMP [0] (1065760483) Thu Oct 09 21:34:43 2003
1  PIN_FLD_TOTAL          ARRAY [840] allocated 20, used 1
2  PIN_FLD_AMOUNT         DECIMAL [0] 0.0166667
1  PIN_FLD_BAL_IMPACTS    ARRAY [0] allocated 20, used 17
2  PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 13
2  PIN_FLD_AMOUNT         DECIMAL [0] 0.0166667
2  PIN_FLD_RESOURCE_ID    INT [0] 840
2  PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 10030 0
2  PIN_FLD_RATE_OBJ       POID [0] 0.0.0.1 /rate 9390 1
2  PIN_FLD_DISCOUNT      DECIMAL [0] 0
2  PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2  PIN_FLD_GL_ID          INT [0] 104
2  PIN_FLD_IMPACT_TYPE     ENUM [0] 1
2  PIN_FLD_QUANTITY       DECIMAL [0] 60.00000000
2  PIN_FLD_RATE_TAG       STR [0] "$1 per hour"
2  PIN_FLD_TAX_CODE       STR [0] ""
2  PIN_FLD_IMPACT_CATEGORY STR [0] "default"
2  PIN_FLD_PACKAGE_ID     INT [0] 20030910
2  PIN_FLD_LINEAGE        STR [0] ""
2  PIN_FLD_PERCENT        DECIMAL [0] 1
2  PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1  PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
0 PIN_FLD_DISCOUNTS     ARRAY [0] allocated 20, used 8
1  PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 0
1  PIN_FLD_OWNER_OBJ      POID [0] 0.0.0.1 /service/ip 11907 1
1  PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1  PIN_FLD_DISCOUNT_LIST ARRAY [0] allocated 20, used 19
2  PIN_FLD_CREATED_T      TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2  PIN_FLD_CYCLE_END_T    TSTAMP [0] (0) <null>
2  PIN_FLD_CYCLE_START_T  TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2  PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 0 0
2  PIN_FLD_DESCR          STR [0] ""
2  PIN_FLD_DISCOUNT_OBJ  POID [0] 0.0.0.1 /discount 8273 0
2  PIN_FLD_LAST_MODIFIED_T TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2  PIN_FLD_PACKAGE_ID     INT [0] 12222
2  PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.0 0 0
2  PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2  PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2  PIN_FLD_QUANTITY       DECIMAL [0] 1
2  PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.0 0 0
2  PIN_FLD_STATUS         ENUM [0] 1
2  PIN_FLD_STATUS_FLAGS   INT [0] 1
2  PIN_FLD_USAGE_END_T    TSTAMP [0] (0) <null>
2  PIN_FLD_USAGE_START_T  TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2  PIN_FLD_FLAGS          INT [0] 1
2  PIN_FLD_TYPE           ENUM [0] 602
1  PIN_FLD_DISCOUNT_LIST ARRAY [1] allocated 20, used 19
2  PIN_FLD_CREATED_T      TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2  PIN_FLD_CYCLE_END_T    TSTAMP [0] (1071385462) Sat Dec 13 23:04:22 2003
2  PIN_FLD_CYCLE_START_T  TSTAMP [0] (1052895862) Wed May 14 00:04:22 2003
2  PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 0 0

```

```

2     PIN_FLD_DESCR          STR [0] ""
2     PIN_FLD_DISCOUNT_OBJ POID [0] 0.0.0.1 /discount 11345 0
2     PIN_FLD_LAST_MODIFIED_T TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2     PIN_FLD_PACKAGE_ID    INT [0] 22222
2     PIN_FLD_PLAN_OBJ      POID [0] 0.0.0.0 0 0
2     PIN_FLD_PURCHASE_END_T TSTAMP [0] (1068793462) Thu Nov 13 23:04:22 2003
2     PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2     PIN_FLD_QUANTITY      DECIMAL [0] 1
2     PIN_FLD_SERVICE_OBJ   POID [0] 0.0.0.1 /service/ip 11907 1
2     PIN_FLD_STATUS        ENUM [0] 1
2     PIN_FLD_STATUS_FLAGS  INT [0] 1
2     PIN_FLD_USAGE_END_T   TSTAMP [0] (1068793462) Thu Nov 13 23:04:22 2003
2     PIN_FLD_USAGE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2     PIN_FLD_FLAGS        INT [0] 1
2     PIN_FLD_TYPE          ENUM [0] 602
1     PIN_FLD_DISCOUNT_LIST ARRAY [2] allocated 28, used 28
2     PIN_FLD_POID          POID [0] 0.0.0.1 /discount 8219 1
2     PIN_FLD_CREATED_T     TSTAMP [0] (1064333980) Tue Sep 23 09:19:40 2003
2     PIN_FLD_MOD_T        TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2     PIN_FLD_READ_ACCESS   STR [0] "B"
2     PIN_FLD_WRITE_ACCESS  STR [0] "S"
2     PIN_FLD_ACCOUNT_OBJ   POID [0] 0.0.0.1 /account 1 1
2     PIN_FLD_DESCR        STR [0] ""
2     PIN_FLD_END_T        TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2     PIN_FLD_MODE         ENUM [0] 801
2     PIN_FLD_NAME         STR [0] "System discount 1"
2     PIN_FLD_OWN_MAX      DECIMAL [0] 0
2     PIN_FLD_OWN_MIN      DECIMAL [0] 0
2     PIN_FLD_PERMITTED    STR [0] ""
2     PIN_FLD_PRIORITY     DECIMAL [0] 1
2     PIN_FLD_PURCHASE_MAX DECIMAL [0] 0
2     PIN_FLD_PURCHASE_MIN DECIMAL [0] 0
2     PIN_FLD_START_T      TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2     PIN_FLD_TYPE         ENUM [0] 603
2     PIN_FLD_USAGE_MAP    ARRAY [0] allocated 20, used 4
3         PIN_FLD_DISCOUNT_MODEL STR [0] "DMStandard"
3         PIN_FLD_EVENT_TYPE  STR [0] "/event"
3         PIN_FLD_FLAGS      INT [0] 0
3         PIN_FLD_SNOWBALL_FLAG INT [0] 0
2     PIN_FLD_DISCOUNT_OBJ POID [0] 0.0.0.1 /discount 8219 1
2     PIN_FLD_SERVICE_OBJ   POID [0] NULL poid pointer
2     PIN_FLD_PACKAGE_ID    INT [0]
2     PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2     PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2     PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2     PIN_FLD_USAGE_END_T   TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2     PIN_FLD_STATUS        ENUM [0] 1
2     PIN_FLD_FLAGS        INT [0] 1
1     PIN_FLD_DISCOUNT_LIST ARRAY [3] allocated 28, used 28
2     PIN_FLD_POID          POID [0] 0.0.0.1 /discount 9755 1
2     PIN_FLD_CREATED_T     TSTAMP [0] (1064334036) Tue Sep 23 09:20:36 2003
2     PIN_FLD_MOD_T        TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2     PIN_FLD_READ_ACCESS   STR [0] "B"
2     PIN_FLD_WRITE_ACCESS  STR [0] "S"
2     PIN_FLD_ACCOUNT_OBJ   POID [0] 0.0.0.1 /account 1 1
2     PIN_FLD_DESCR        STR [0] ""
2     PIN_FLD_END_T        TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2     PIN_FLD_MODE         ENUM [0] 801
2     PIN_FLD_NAME         STR [0] "Sys discount 3"
2     PIN_FLD_OWN_MAX      DECIMAL [0] 0

```

```

2      PIN_FLD_OWN_MIN      DECIMAL [0] 0
2      PIN_FLD_PERMITTED    STR [0] ""
2      PIN_FLD_PRIORITY     DECIMAL [0] 14
2      PIN_FLD_PURCHASE_MAX DECIMAL [0] 0
2      PIN_FLD_PURCHASE_MIN DECIMAL [0] 0
2      PIN_FLD_START_T      TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_TYPE         ENUM [0] 603
2      PIN_FLD_USAGE_MAP    ARRAY [0] allocated 20, used 4
3          PIN_FLD_DISCOUNT_MODEL STR [0] "DMStandard"
3          PIN_FLD_EVENT_TYPE STR [0] "/event/session"
3          PIN_FLD_FLAGS     INT [0] 1
3          PIN_FLD_SNOWBALL_FLAG INT [0] 0
2      PIN_FLD_DISCOUNT_OBJ POID [0] 0.0.0.1 /discount 9755 1
2      PIN_FLD_SERVICE_OBJ   POID [0] NULL poid pointer
2      PIN_FLD_PACKAGE_ID    INT [0]
2      PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2      PIN_FLD_USAGE_END_T   TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2      PIN_FLD_STATUS        ENUM [0] 1
2      PIN_FLD_FLAGS         INT [0] 1
1  PIN_FLD_DISCOUNT_LIST  ARRAY [4] allocated 28, used 28
2      PIN_FLD_POID          POID [0] 0.0.0.1 /discount 11291 1
2      PIN_FLD_CREATED_T     TSTAMP [0] (1064334029) Tue Sep 23 09:20:29 2003
2      PIN_FLD_MOD_T         TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_READ_ACCESS   STR [0] "B"
2      PIN_FLD_WRITE_ACCESS  STR [0] "S"
2      PIN_FLD_ACCOUNT_OBJ   POID [0] 0.0.0.1 /account 1 1
2      PIN_FLD_DESCR        STR [0] ""
2      PIN_FLD_END_T         TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2      PIN_FLD_MODE          ENUM [0] 801
2      PIN_FLD_NAME          STR [0] "Sys discount 2"
2      PIN_FLD_OWN_MAX       DECIMAL [0] 0
2      PIN_FLD_OWN_MIN       DECIMAL [0] 0
2      PIN_FLD_PERMITTED     STR [0] ""
2      PIN_FLD_PRIORITY      DECIMAL [0] 200
2      PIN_FLD_PURCHASE_MAX  DECIMAL [0] 0
2      PIN_FLD_PURCHASE_MIN  DECIMAL [0] 0
2      PIN_FLD_START_T       TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_TYPE          ENUM [0] 603
2      PIN_FLD_USAGE_MAP     ARRAY [0] allocated 20, used 4
3          PIN_FLD_DISCOUNT_MODEL STR [0] "DMStandard"
3          PIN_FLD_EVENT_TYPE STR [0] "/event/session"
3          PIN_FLD_FLAGS     INT [0] 1
3          PIN_FLD_SNOWBALL_FLAG INT [0] 0
2      PIN_FLD_DISCOUNT_OBJ POID [0] 0.0.0.1 /discount 11291 1
2      PIN_FLD_SERVICE_OBJ   POID [0] NULL poid pointer
2      PIN_FLD_PACKAGE_ID    STR [0]
2      PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2      PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2      PIN_FLD_USAGE_END_T   TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2      PIN_FLD_STATUS        ENUM [0] 1
2      PIN_FLD_FLAGS         INT [0] 1
0  PIN_FLD_BAL_INFO        ARRAY [0] allocated 20, used 3
1      PIN_FLD_BAL_GRP_OBJ   POID [0] 0.0.0.1 /balance_group 8323 4
1      PIN_FLD_BALANCES      ARRAY [840] allocated 11, used 6
2          PIN_FLD_NEXT_BAL   DECIMAL [0] 0
2          PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
2          PIN_FLD_CURRENT_BAL DECIMAL [0] 19.590836

```

```

2     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 100
2     PIN_FLD_CREDIT_FLOOR DECIMAL [0] 0
2     PIN_FLD_CREDIT_THRESHOLDS INT [0] 0
1     PIN_FLD_BALANCES ARRAY [1000001] allocated 7, used 6
2     PIN_FLD_NEXT_BAL DECIMAL [0] 0
2     PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
2     PIN_FLD_CURRENT_BAL DECIMAL [0] 0
2     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 100
2     PIN_FLD_CREDIT_FLOOR DECIMAL [0] 0
2     PIN_FLD_CREDIT_THRESHOLDS INT [0] 0
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1     PIN_FLD_POID POID [0] 0.0.0.1 /account 10243 13
1     PIN_FLD_MOD_T TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
1     PIN_FLD_ACCOUNT_NO STR [0] "0.0.0.1-10243"
1     PIN_FLD_BRAND_OBJ POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_TIMEZONE_ID STR [0] ""
1     PIN_FLD_STATUS ENUM [0] 10100
1     PIN_FLD_STATUS_FLAGS INT [0] 0
1     PIN_FLD_CURRENCY INT [0] 840
1     PIN_FLD_CURRENCY_SECONDARY INT [0] 0
1     PIN_FLD_GROUP_OBJ POID [0] 0.0.0.0 0 0
1     PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
1     PIN_FLD_ITEM_POID_LIST STR [0] "0.0.0.1|/item/misc 8835 0"
1     PIN_FLD_NEXT_ITEM_POID_LIST STR [0] ""
1     PIN_FLD_ACTG_TYPE ENUM [0] 2
1     PIN_FLD_LAST_STATUS_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_GL_SEGMENT STR [0] "."
1     PIN_FLD_BILL_WHEN INT [0] 1
1     PIN_FLD_PAY_TYPE ENUM [0] 10001
1     PIN_FLD_AR_BILLINFO_OBJ POID [0] 0.0.0.1 /billinfo 8451 0
1     PIN_FLD_NEXT_BILL_OBJ POID [0] 0.0.0.0 0 0
1     PIN_FLD_NEXT_BILL_T TSTAMP [0] (1065769200) Fri Oct 10 00:00:00 2003
1     PIN_FLD_LAST_BILL_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_ACTG_LAST_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_ACTG_FUTURE_T TSTAMP [0] (1068451200) Mon Nov 10 00:00:00 2003
1     PIN_FLD_BILL_ACTGCYCLES_LEFT INT [0] 1
1     PIN_FLD_PAYINFO_OBJ POID [0] 0.0.0.1 /payinfo/invoice 11267 0
1     PIN_FLD_ACTG_NEXT_T TSTAMP [0] (1065769200) Fri Oct 10 00:00:00 2003
1     PIN_FLD_LAST_BILL_OBJ POID [0] 0.0.0.0 0 0
1     PIN_FLD_BILL_OBJ POID [0] 0.0.0.1 /bill 10499 0
1     PIN_FLD_PENDING_RECV DECIMAL [0] 0
1     PIN_FLD_BAL_GRP_OBJ POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_SERVICE_INFO SUBSTRUCT [0] allocated 51, used 26
2     PIN_FLD_POID POID [0] 0.0.0.1 /service/ip 11907 5
2     PIN_FLD_CREATED_T TSTAMP [0] (1063217471) Wed Sep 10 11:11:11 2003
2     PIN_FLD_MOD_T TSTAMP [0] (1063217473) Wed Sep 10 11:11:13 2003
2     PIN_FLD_READ_ACCESS STR [0] "L"
2     PIN_FLD_WRITE_ACCESS STR [0] "L"
2     PIN_FLD_AAC_ACCESS STR [0] ""
2     PIN_FLD_AAC_PACKAGE STR [0] ""
2     PIN_FLD_AAC_PROMO_CODE STR [0] ""
2     PIN_FLD_AAC_SERIAL_NUM STR [0] ""
2     PIN_FLD_AAC_SOURCE STR [0] ""
2     PIN_FLD_AAC_VENDOR STR [0] ""
2     PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 10243 0
2     PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
2     PIN_FLD_EFFECTIVE_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
2     PIN_FLD_ITEM_POID_LIST STR [0] "0.0.0.1|/item/cycle_forward 11651 0"
2     PIN_FLD_LASTSTAT_CMNT STR [0] ""
2     PIN_FLD_LAST_STATUS_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003

```

```

2      PIN_FLD_LOGIN          STR [0] "00491732411"
2      PIN_FLD_NAME           STR [0] "PIN Service Object"
2      PIN_FLD_NEXT_ITEM_POID_LIST STR [0] ""
2      PIN_FLD_PASSWD         STR [0] "clear|00491732411"
2      PIN_FLD_PROFILE_OBJ     POID [0] 0.0.0.0 0 0
2      PIN_FLD_STATUS         ENUM [0] 10100
2      PIN_FLD_STATUS_FLAGS    INT [0] 0
2      PIN_FLD_SERVICE_IP     SUBSTRUCT [0] allocated 20, used 3
3          PIN_FLD_COMPRESSION ENUM [0] 0
3          PIN_FLD_IPADDR      BINSTR [0] 1 00
3          PIN_FLD_PROTOCOL    ENUM [0] 0
2      PIN_FLD_BAL_GRP_OBJ     POID [0] 0.0.0.1 /balance_group 8323 4

```

Sample Fields Required by Pipeline Manager

The following are sample fields, in flist format, required by Pipeline Manager when the event type is **/event/session**:

Important: You cannot trim the default fields for the PIN_FLD_INHERITED_INFO substruct listed in ["Sample Unmodified Flist"](#). However, you can specify additional **/account** and **/service** fields. In the text below, the **/account** field PIN_FLD_RESIDENCE_FLAG is specified at the end of the list. It is added to the default PIN_FLD_INHERITED_INFO fields sent to Pipeline Manager.

```

0 PIN_FLD_POID                POID [0] 0.0.0.1 /event/session -1 0
0 PIN_FLD_EVENT               SUBSTRUCT [0] allocated 25, used 25
1     PIN_FLD_POID             POID [0] 0.0.0.1 /event/session -1 0
1     PIN_FLD_START_T          TSTAMP [0] (1065785673) Fri Oct 10 04:34:33 2003
1     PIN_FLD_END_T            TSTAMP [0] (1065785683) Fri Oct 10 04:34:43 2003
1     PIN_FLD_BAL_IMPACTS      ARRAY [0] allocated 20, used 17 and other array elements
2         PIN_FLD_AMOUNT        DECIMAL [0] 0.0166667
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_RESOURCE_ID    INT [0] 840
2         PIN_FLD_GL_ID          INT [0] 104
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 1
2         PIN_FLD_QUANTITY       DECIMAL [0] 60.00000000
2         PIN_FLD_RATE_TAG       STR [0] "$1 per hour"
2         PIN_FLD_TAX_CODE       STR [0] ""
0 PIN_FLD_DISCOUNTS         ARRAY [0] allocated 20, used 8 and other array elements
1     PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 10243 0
1     PIN_FLD_OWNER_OBJ        POID [0] 0.0.0.1 /service/ip 11907 1
1     PIN_FLD_BAL_GRP_OBJ      POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_DISCOUNT_LIST   ARRAY [0] allocated 20, used 19 and other array elements
2         PIN_FLD_DISCOUNT_OBJ POID [0] 0.0.0.1 /discount 8273 0
2         PIN_FLD_PACKAGE_ID     INT [0] 12222
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_QUANTITY       DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_USAGE_END_T     TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_T   TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_FLAGS          INT [0] 1
2         PIN_FLD_TYPE           ENUM [0] 602
0 PIN_FLD_BAL_INFO           ARRAY [0] allocated 20, used 3 and other array elements
1     PIN_FLD_BAL_GRP_OBJ      POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_BALANCES         ARRAY [840] allocated 11, used 6 and other array elements
2         PIN_FLD_CURRENT_BAL    DECIMAL [0] 19.590836

```

```

1   PIN_FLD_BALANCES      ARRAY [1000001] allocated 7, used 6 and other array elements
2       PIN_FLD_CURRENT_BAL DECIMAL [0] 0
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1   PIN_FLD_RESIDENCE_FLAG ENUM [0] 1

```

A different set of fields is required when the event type is `/event/session/` (including the final forward slash), and another set of fields is sent for any other type of event.

To implement the trimmed flist in the example, create the following XML file (**sample.xml**). When this XML file is loaded with **load_pin_rtp_trim_flist**, the flist sent to Pipeline Manager is constructed as follows:

- If the event type is exactly `/event/session/`, the `PIN_FLD_RESIDENCE_FLAG` field is included with the trimmed flist as shown in the flist sample above.
- If the event type starts with `/event/session/` (including the last forward slash), the `PIN_FLD_RESIDENCE_FLAG` field is not included with the trimmed flist.
- If the event type is any other value (which matches the section specified by **Type** value * with **Flags** value 1), then neither the `PIN_FLD_RESIDENCE_FLAG` field nor the `PIN_FLD_BAL_IMPACTS` array is included with the trimmed flist.

Important: You cannot trim the default fields for the `PIN_FLD_INHERITED_INFO` substruct listed in ["Sample Unmodified Flist"](#). However, you can specify additional `/account` and `/service` fields. In the text below, the `/account` field `PIN_FLD_RESIDENCE_FLAG` is specified at the end of the list. It is added to the default `PIN_FLD_INHERITED_INFO` fields sent to Pipeline Manager.

sample.xml File

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
=====
Copyright (c) 2004 Portal Software, Inc. All rights reserved.
This material is the confidential property of Portal Software, Inc.
or its Subsidiaries or licensors and may be used, reproduced, stored
or transmitted only in accordance with a valid Portal license or
sublicense agreement.
=====
-->

<RTPTrimFlistConfiguration xmlns="http://www.portal.com/InfranetXMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.portal.com/InfranetXMLSchema pin_config_rtp_trim_flist.xsd">
  <EventMapList>
    <EventMap>

<!-- Section which specifies fields sent when the event type is exactly /event/session -->

      <Event>
        <Type>/event/session</Type>
        <Flags>0</Flags>
      </Event>
      <RequiredField>
        <Name>PIN_FLD_POID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_EVENT</Name>

```

```

<RequiredField>
  <Name>PIN_FLD_POID</Name>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_START_T</Name>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_END_T</Name>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_BAL_IMPACTS</Name>
  <RequiredField>
    <Name>PIN_FLD_AMOUNT</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_AMOUNT_DEFERRED</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_RESOURCE_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_GL_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_IMPACT_TYPE</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_QUANTITY</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_RATE_TAG</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_TAX_CODE</Name>
  </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_DISCOUNTS</Name>
  <RequiredField>
    <Name>PIN_FLD_ACCOUNT_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_OWNER_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_BAL_GRP_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_DISCOUNT_LIST</Name>
    <RequiredField>
      <Name>PIN_FLD_DISCOUNT_OBJ</Name>
    </RequiredField>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PACKAGE_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PURCHASE_END_T</Name>
  </RequiredField>
  <RequiredField>

```



```

        <Name>PIN_FLD_PURCHASE_START_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_QUANTITY</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_STATUS</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_USAGE_END_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_USAGE_START_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_FLAGS</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_TYPE</Name>
    </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
    <Name>PIN_FLD_BAL_INFO</Name>
    <RequiredField>
        <Name>PIN_FLD_BAL_GRP_OBJ</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_BALANCES</Name>
    <RequiredField>
        <Name>PIN_FLD_CURRENT_BAL</Name>
    </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
    <Name>PIN_FLD_INHERITED_INFO</Name>
    <RequiredField>
        <Name>PIN_FLD_RESIDENCE_FLAG</Name>
    </RequiredField>
</RequiredField>
</EventMap>

```

<!-- Section which specifies fields sent when the event type starts with /event/session/ -->

```

</EventMap>
<Event>
    <Type>/event/session/</Type>
    <Flags>1</Flags>
</Event>
<RequiredField>
    <Name>PIN_FLD_POID</Name>
</RequiredField>
<RequiredField>
    <Name>PIN_FLD_EVENT</Name>
    <RequiredField>
        <Name>PIN_FLD_POID</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_START_T</Name>
    </RequiredField>

```

```
<RequiredField>
  <Name>PIN_FLD_END_T</Name>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_BAL_IMPACTS</Name>
  <RequiredField>
    <Name>PIN_FLD_AMOUNT</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_RESOURCE_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_GL_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_IMPACT_TYPE</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_QUANTITY</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_RATE_TAG</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_TAX_CODE</Name>
  </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_DISCOUNTS</Name>
  <RequiredField>
    <Name>PIN_FLD_ACCOUNT_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_OWNER_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_BAL_GRP_OBJ</Name>
  </RequiredField>
<RequiredField>
  <Name>PIN_FLD_DISCOUNT_LIST</Name>
  <RequiredField>
    <Name>PIN_FLD_DISCOUNT_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PACKAGE_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PURCHASE_END_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PURCHASE_START_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_QUANTITY</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_STATUS</Name>
  </RequiredField>
</RequiredField>
```

```

        <Name>PIN_FLD_USAGE_END_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_USAGE_START_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_FLAGS</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_TYPE</Name>
    </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
    <Name>PIN_FLD_BAL_INFO</Name>
    <RequiredField>
        <Name>PIN_FLD_BAL_GRP_OBJ</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_BALANCES</Name>
    <RequiredField>
        <Name>PIN_FLD_CURRENT_BAL</Name>
    </RequiredField>
</RequiredField>
</RequiredField>
</EventMap>

```

<!--* Section which specifies fields sent when the event type is
 * any other value.-->

```

<EventMap>
    <Event>
        <Type>*</Type>
        <Flags>1</Flags>
    </Event>
    <RequiredField>
        <Name>PIN_FLD_POID</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_EVENT</Name>
    <RequiredField>
        <Name>PIN_FLD_POID</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_START_T</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_END_T</Name>
    </RequiredField>
</RequiredField>
<RequiredField>
    <Name>PIN_FLD_DISCOUNTS</Name>
    <RequiredField>
        <Name>PIN_FLD_ACCOUNT_OBJ</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_OWNER_OBJ</Name>
    </RequiredField>
    <RequiredField>
        <Name>PIN_FLD_BAL_GRP_OBJ</Name>
    </RequiredField>

```

```
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_DISCOUNT_LIST</Name>
  <RequiredField>
    <Name>PIN_FLD_DISCOUNT_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PACKAGE_ID</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PURCHASE_END_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_PURCHASE_START_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_QUANTITY</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_STATUS</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_USAGE_END_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_USAGE_START_T</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_FLAGS</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_TYPE</Name>
  </RequiredField>
</RequiredField>
</RequiredField>
<RequiredField>
  <Name>PIN_FLD_BAL_INFO</Name>
  <RequiredField>
    <Name>PIN_FLD_BAL_GRP_OBJ</Name>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_BALANCES</Name>
    <RequiredField>
      <Name>PIN_FLD_CURRENT_BAL</Name>
    </RequiredField>
  </RequiredField>
</RequiredField>
</EventMap>
</EventMapList>
</RTPTrimFlistConfiguration>
```

Measuring System Latencies with Instrumentation

You use the Pipeline Manager instrumentation feature to determine how much processing time each Pipeline Manager component (function modules, iScripts, and iRules) is consuming in microseconds. This information enables you to:

- Determine system benchmarks.
- Identify performance bottlenecks at the function module level.

- Add or reconfigure function pools to optimize CPU utilization.

Instrumentation collects statistics for the following components:

- The input module.
- Each function module.
- The output module.

After each transaction, the statistics for each pipeline tested are written to the **pipeline.log** file.

Using Instrumentation to Collect Module Performance Statistics

To enable instrumentation:

1. Start the pipeline.
2. Send a signal to the pipeline to toggle instrumentation on and off. Use the following commands to toggle the instrumentation state:

Solaris, Linux, and AIX:

```
kill -s USR1 ifw_process_pid
```

HP-UX IA64:

```
kill -USR1 ifw_process_pid
```

AIX:

```
kill -s USR1 ifw_process_pid
```

At the end of each transaction, the statistics are logged to the **pipeline.log** file and the statistics counters are reset.

Note: By default, Pipeline Manager instrumentation is disabled on startup. When Pipeline Manager is running, you can toggle between the disabled and enabled modes.

Important: Pipeline Manager begins gathering statistics immediately after receiving the signal. To assure accurate measurements, be sure that Pipeline Manager is not processing transactions when the signal is sent.

3. Process a sample CDR file.
4. Check the pipeline log files for processing time statistics. See "[Viewing Instrumentation Testing Results](#)" for more information.
5. When testing is complete, stop the instrumentation process by sending another signal. See step 2.

Viewing Instrumentation Testing Results

Each log file record consists of the fully qualified module name and the accumulated processing time spent in the module.

Note: Pipeline processing time statistics are not cumulative. The output module writes data to a file whereas a function module processes EDRs in a different thread.

Sample Log File

The following sample log file shows instrumentation data:

```
15.03.2004 13:25:07 test      ifw      IFW      NORMAL  00516 - (ifw.Pipelines.ALL_
RATE.Functions.PreProcessing) Plugin processing time statistics: 'ifw.Pipelines.ALL_
RATE.Functions.PreProcessing.FunctionPool.APNMap
.Module, 7676390
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.CustomerRating.Module, 22629863
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.CustomerSearch.Module, 239523272
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.EventDiscarding.Module, 19874050
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PreRatingZone.Module, 18751824
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PreRecycle.Module, 1916139
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PrefixDesc.Module, 8001348
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.ServiceCodeMap.Module, 4543899
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageClassMap.Module, 6083775
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageScenarioMap.Module, 9786078
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageType.Module, 57114053
,
15.03.2004 13:25:07 test      ifw      IFW      NORMAL  00516 - (ifw.Pipelines.ALL_
RATE.Functions.Rating) Plugin processing time statistics: 'ifw.Pipelines.ALL_
RATE.Functions.Rating.FunctionPool.AddInfranetBillingRe
cord.Module, 44927730
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.MainRating.Module, 78250224
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.RateAdjustment.Module, 2358093
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.Recycle.Module, 1225628
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.Rejection.Module, 1785748
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.ServiceOutputSplit.Module, 6480466
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.SpecialDayRate.Module, 8109825
,
15.03.2004 13:25:07 test      ifw      IFW      NORMAL  00516 - (ifw.Pipelines.ALL_
RATE.Output.OutputCollection) Plugin processing time statistics: 'ifw.Pipelines.ALL_
RATE.Output.OutputCollection.DevNull.Module, 67
ifw.Pipelines.ALL_RATE.Output.OutputCollection.DuplicateOutput.Module, 23
ifw.Pipelines.ALL_RATE.Output.OutputCollection.FAXOutput.Module, 561
ifw.Pipelines.ALL_RATE.Output.OutputCollection.GPRSOutput.Module, 728
ifw.Pipelines.ALL_RATE.Output.OutputCollection.RejectOutput.Module, 30
ifw.Pipelines.ALL_RATE.Output.OutputCollection.SMSOutput.Module, 552
ifw.Pipelines.ALL_RATE.Output.OutputCollection.TELOutput.Module, 178434585
ifw.Pipelines.ALL_RATE.Output.OutputCollection.WAPOutput.Module, 550
Note: To aggregate the counters into a nice report, please take a look at TracePerformanceReporting
```

Optimizing the DAT_USC_Map Module

The DAT_USC_Map module uses the Pipeline Manager framework component (FSM) to compile data mapping rules, which are stored in the database as regular expressions. The FSM compiles the data mapping structures only during Pipeline Manager startup because the rules can contain many comparisons of mapping patterns; this impacts startup performance. You can optimize the DAT_USC_Map module to enable Pipeline Manager to serialize the data structures and restore them from the serialized format.

About Precompiling USC Mapping Rules

Not all USC mapping data is stored in a compiled format: for example, rules used to define zone models. When the DAT_USC_Map module loads event data, it reorganizes it according to zone models to enable faster searching of the data structures during run time. This increases load time and memory requirements. To reduce the impact, you can configure Pipeline Manager to serialize the data structures the first time they are loaded and then reuse the serialized version during subsequent startup operations.

When Pipeline Manager begins processing data for a given zone model, it checks to see if a precompiled data file exists for that zone model. If so, it prepares the complex data structure by using the serialized format rather than by recompiling the structure from the USC map data.

If you enable the precompiling functionality, the following data is serialized:

- USC group
- Usage class and usage type
- Service code and service class
- Wholesale zone and retail zone

Note: Data that is not in the precompiled format is read from the database or file system, depending on your DAT_USC_Map module configuration. See "DAT_USC_Map" in *BRM Configuring Pipeline Rating and Discounting*.

For more information, see ["Precompiling Usage Scenario Mapping Data"](#).

About Filtering Mapping Rules

You use USC groups to assemble the rules that define which services and service configurations are available to the pipeline; they contain the rules for mapping the service EDR attributes to each usage class.

You can configure your system to filter mapping rules based on USC groups so only the rules in the USC groups you specify are compiled and loaded into the DAT_USC_Map module. All other rules are ignored. This is more efficient than having one zone model that uses a large number of rules.

Note: This is necessary only when your USC mapping rules are stored in the database; if they are read from a file, the data is already organized according to USC groups.

Generally you define USC Groups to contain the mapping rules for a specific type of EDR processing. For example, say you rate telephony services and process EDRs by using three USC groups (GSM, SMS, and GPRS), each of which contains mapping rules to determine domestic standard charges, domestic roaming charges, and international charges.

To increase performance, you can define the mapping rules for each set of charges in a separate zone model. Then, when an EDR is processed, based on the USC group specified, only the rules used in those zone models are compiled and loaded. This increases startup performance.

For more information, see ["Filtering the Mapping Data to Compile and Load"](#).

For information on USC groups, see "About Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.

Configuring the DAT_USC_Map Module for Startup Performance

You improve startup performance of the DAT_USC_Map module by:

- Increasing the number of threads used to load mapping data.
- Precompiling usage scenario mapping data.
- Filtering the mapping data to compile and load.

You define these configurations in the Pipeline Manager registry file. For more information, see ["About Configuring Pipeline Manager"](#).

Increasing the Number of Threads Used to Load Mapping Data

The DAT_USC_Map module loads mapping rules for each zone model in a USC group by using a separate thread; therefore, it is only necessary to increase the number of threads when your USC groups contain multiple zone models.

To use multiple threads, set the **NumberOfThreads** registry entry to the desired number of threads. This enables Pipeline Manager to compile data in parallel and to restore it from the precompiled data files.

For example:

NumberOfThreads = 4

The default is 1.

Note: You can use this entry as a semaphore.

Precompiling Usage Scenario Mapping Data

To enable precompiling of USC mapping data, set the **PreCompiledDataDir** registry entry. This entry both enables the precompile functionality and defines the location of the compiled data files. By default, compiled data files are saved in the **.compiled_usc_data** directory.

Pipeline Manager saves them with the following naming convention:

USCzoneModelID.pc

For example, **GSM.pc**, **GSM_DOMESTIC.pc**, and **GSM_ROAMING.pc**.

If this entry is set, compiled files are created the next time the pipeline starts. For each subsequent run, the data files are validated against the data structures in the database and, if necessary, recompiled and resaved to the file system.

Note: You can use this entry as a semaphore.

Filtering the Mapping Data to Compile and Load

If the source for your USC mapping rules is the database rather than a file, you can filter which rules are compiled and loaded into the DAT_USC_Map module when a

pipeline starts by setting the **UscGroups** registry entry to one or more USC groups. For example:

```
UscGroups {GSM GSM_ROAMING}
```

Important: You can specify only one USC group per each pipeline running in your system. If you use multiple USC groups, you must configure Pipeline Manager to run multiple pipeline instances. To do this, configure the Pipeline Manager registry file so the FCT_USC_Map in each pipeline instance refers to the appropriate DAT_USC_Map module reference and **UscGroups** entry. For more information, see "[About Configuring Pipeline Manager](#)".

By default, all mapping rules are loaded into the pipeline. see "[About Filtering Mapping Rules](#)" for more information.

Note: You can use this entry as a semaphore.

Using Semaphores

You can use the new **NumberOfThreads**, **PreCompiledDataDir**, and **UscGroups** registry entries as semaphores to configure and control Pipeline Manager during pipeline startup. These semaphores perform the same tasks that the **Reload** semaphore performs, as specified in the startup registry or last-processed semaphore:

1. Load mapping data from the source (**Database** or **File**).
2. Create the USC zone model (from data in **PreCompiledDataDir** or **USCMapFile**).
3. Compile or precompile each USC zone model.

When you change the values of these semaphores after startup, they are not updated automatically in your system; you must use the **Reload** semaphore to update them during run time.

For example:

- To use multiple threads to load data, edit the **NumberOfThreads** semaphore and then call the **Reload** semaphore. Each thread processes a different zone model when loading the USC data.
- To reload USC data using a different set of files in the **PreCompiledDataDir** directory, edit the **PreCompiledDataDir** semaphore and then call the **Reload** semaphore.
- To filter a different set of mapping rules, edit the **UscGroups** semaphore and then call the **Reload** semaphore.

For more information on the DAT_USC_Map semaphore entries, see "Semaphore File Entries" in *BRM Configuring Pipeline Rating and Discounting*.

Other Pipeline Manager Monitoring Tools

This section describes additional Pipeline Manager performance-monitoring tools.

Viewing the Pipeline Log

You can see the results of tests for each pipeline in that pipeline's **pipeline.log** file.

Tip: Open each log file in a terminal windows and run **tail -f** on the logs.

After each batch stream is processed, the pipeline writes the following information to the **pipeline.log** files:

- The number of processed EDRs.
- The number of errors that occurred during EDR processing.
- The number of EDRs processed per second for a stream.
- If instrumentation is on, the instrumentation results. See "[Viewing Instrumentation Testing Results](#)" for more information.

Tuning Tips

- Let the system process a few files before you measure performance. This assures that any additional memory needed (for example, for the buffers) has been allocated.
- Use the system monitor tool to monitor system utilization.

Configuring Buffer Size Polling

Use the **QueueRequestTimeout** Controller entry in the registry to specify the interval in seconds that each queue's fill status is written to the log. For example:

```
ifw
{
    Active = TRUE
    ProcessLoopTimeout = 10
    QueueRequestTimeout = 10 # Optional, 0 disables
    ...
}
```

The default is 0 (no polling).

Buffer fill status information can indicate which function pool is the slowest. Over time, buffers in front of the slowest function pool fill up, and those that occur later in the stream are empty.

Note: Instrumentation is the recommended tool for identifying the slowest function pool. See "[Measuring System Latencies with Instrumentation](#)".

OS-Specific Pipeline Manager Monitoring Tools

This section describes OS-specific tools that you can use to monitor and maintain your Pipeline Manager system.

Solaris Monitoring Tools

This section describes Solaris monitoring tools.

Displaying Thread Details

To display details of the threads within a process, use the **prstat** command:

```
prstat -lmv -p
```

Example output:

```
prstat -lmv -p 22376
  PID USERNAME  USR  SYS  TRP  TFL  DFL  LCK  SLP  LAT  VCX  ICX  SCL  SIG  PROCESS/LWPID
22376 integ      86   13  0.0  0.0  0.0  0.0  0.0  0.9   12   3K   .1M   0  ifw/4
22376 integ      61   34  0.0  0.0  0.0  0.5  2.8  2.0  298   1K   64K   0  ifw/16
22376 integ      52   0.8  0.0  0.0  0.0  42  0.0  4.9   56  11K  11K   0  ifw/117
22376 integ      43   3.6  0.0  0.0  0.0  52  0.0  1.8  158   1K   7K   0  ifw/5
22376 integ      22   0.1  0.0  0.0  0.0  75  0.0  2.6  393  125  463   0  ifw/116
22376 integ      21   0.1  0.0  0.0  0.0  77  0.0  2.6   89  357  412   0  ifw/115
...
Total: 1 processes, 48 lwps, load averages: 4.18, 1.94, 2.38
```

In the pipeline **process.log** file, you can see when a thread is created, its name, and corresponding OS number:

```
09.12.2003 19:54:38 igscoll1    ifw          IFW          NORMAL    00000 -
(ifw.ProcessLog.Module) Thread instance ID '2'; and
Name 'ifw.ProcessLog.Module'.
...
09.12.2003 20:01:31 igscoll1    ifw          IFW          NORMAL    00000 -
(ifw.Pipelines.GSM.Input) Thread instance ID '16'; and
Name 'ifw.Pipelines.GSM.Input'.
...
09.12.2003 21:38:40 igscoll1    ifw          IFW          NORMAL    00000 -
(ifw.Pipelines.GSM.Functions.PreProcessing2) Thread instance ID '135'; and
Name 'ifw.Pipelines.GSM.Functions.PreProcessing2'.
...
```

Dumping the Call Stack of an Active Process

To dump the call stack of an active process, use the **vmstat** and **mpstat pstack** commands.

Identifying Thread Functions

To identify what each thread is used for, such as the pipeline framework, input, output, or function modules, use the **mpstat** command:

```
mpstat pstack
```

Tip: To determine the correlation between pipeline thread names and thread numbers, see the **pipeline.log** file.

HP-UX IA64 Monitoring Tools

Tools useful for monitoring Pipeline Manager on HP-UX IA64 systems include:

- **glance**
- **sar -AM**
- **top**
- **vmstat**
- **iostat**

For more information on these and other tools, see the HP documentation.

Linux Monitoring Tools

Tools useful for monitoring Pipeline Manager on Linux systems include:

- **vmstat**
- **sar**
- **top**
- **pmap**
- **gnome-system-monitor**
- **sysstat package (iostat, mpstat, sadc, and sar)**

AIX Monitoring Tools

Tools useful for monitoring Pipeline Manager on AIX systems include:

- **sar**
- **vmstat**
- **iostat**
- **filemon**
- **topas**
- **trace**
- **svmon**
- **netpmon**

Optimizing BRM for Prepaid and Postpaid Convergence

This chapter describes cache residency distinction, which enables you to optimize your Oracle Communications Billing and Revenue Management (BRM) system for rating both prepaid and postpaid accounts in both real time and batch.

It assumes you have installed the components necessary for rating prepaid and postpaid accounts and have set up real-time and pipeline batch rating. For more information, see "Putting Together Your BRM System" in *BRM Installation Guide*.

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

- Basic BRM concepts. See "Introducing BRM" in *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- Pricing and rating configuration. See "About Creating a Price List" in *BRM Configuring Pipeline Rating and Discounting*.
- In-Memory Database (IMDB) Cache Manager. See ["Using Oracle IMDB Cache Manager"](#).
- Basic database administration concepts. See your database documentation.

About Convergent BRM Systems

A BRM system that handles prepaid and postpaid subscribers and rates events using both real-time and batch processing is referred to as a *convergent system*.

To allow both types of subscriber accounts and provide both types of rating, you must set up a BRM system with prepaid and postpaid real-time and batch rating. Then you must ensure the following behaviors:

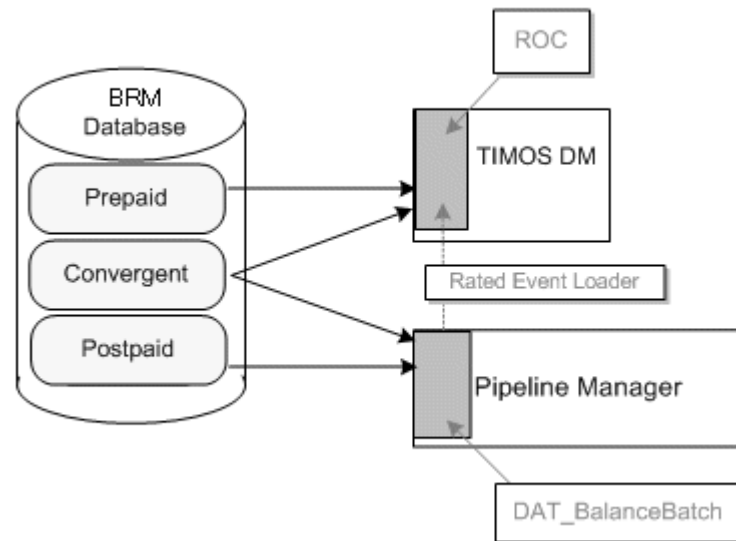
- Only the data relevant to a particular rating system is stored in its cache to improve performance and reduce memory requirements.
- The data stored in or updated by the rating components are synchronized with the database to maintain data integrity.

About Cache Residency

To perform rating, BRM caches subscriber data, such as data from `/account`, `/billinfo`, and `/service` objects as shown in [Figure 16-1](#). Objects associated with prepaid subscribers are stored in IMDB Cache. Objects associated with postpaid subscribers are stored in Pipeline Manager memory. Objects associated with convergent subscribers are stored in both caches.

You can configure your BRM system so only the required objects are loaded into the appropriate memory caches after the object has been created or changed in the database or in one of the caches. This reduces the load time during initialization and data synchronization operations and minimizes memory size.

Figure 16–1 BRM Cache Residency



After events are processed and objects are recorded in the database, the Pipeline Manager memory and IMDB Cache are updated with the data from the database to maintain data integrity. The caches must be synchronized with one another to handle objects that reside in both caches.

Note: Currently, cache residency is supported in Pipeline Manager only for the DAT_BalanceBatch module. For more information about DAT_BalanceBatch module, see *BRM Configuring Pipeline Rating and Discounting*.

How Cache Residency Is Determined

Objects in an IMDB Cache-enabled system have a RESIDENCY_TYPE attribute that identifies how they reside in the BRM system. This value is defined in the data dictionary. [Table 16–1](#) lists the attribute values that identify object types and their residency.

Table 16–1 Object Residency and Attributes

Attribute Value	Object Type	Residency
0	Database objects	Owned by the BRM database.
1	In-memory objects	Owned by IMDB Cache DM.
2	Shared objects	Owned by IMDB Cache DM and the BRM database.
3	Shared objects	Control objects that are passed through IMDB Cache DM.
4	Deferred database objects	Objects owned by the database but passed through IMDB Cache DM.

Table 16–1 (Cont.) Object Residency and Attributes

Attribute Value	Object Type	Residency
5	Static reference objects	Objects owned by the database and cached in IMDB Cache when the flow back from the database.
6	Volatile objects	Objects stored only in IMDB Cache and not persistent.
7	Dynamic reference object	Reference objects that are owned by the database and are updated more frequently in IMDB Cache.

BRUM uses the residency type value to determine which objects to retrieve during request operations. Only reference objects (RESIDENCY_TYPE value of 5 or 7) are valid for caching in memory.

Objects that are updated during batch rating are cached in the pipeline DAT_BalanceBatch module, and objects that are updated during real-time rating are cached in IMDB Cache. The database is updated with all data changed during both real-time and batch processing.

You can configure which objects are eligible for which memory cache when they are updated by associating them with prepaid, postpaid, or convergent business profiles and setting up validation rules.

The following objects are valid for cache residency distinction:

- /account
- /balance_group
- /billinfo
- /group
- /group/sharing
- /group/sharing/charges
- /group/sharing/discounts
- /ordered_balgrp
- /profile
- /profile/acct_extrating
- /profile/serv_extrating
- /purchased_discount
- /purchased_product
- /service
- /uniqueness

Note: You cannot configure the **/account** object or the **/uniqueness** object with a specific cache type. Account object instances receive their cache type values from their associated **/billinfo** object, and **/uniqueness** object instances receive their cache type values from the associated services.

For information on how to configure nonreference objects for cache residency distinction, see "[Configuring Nonreference Objects for Cache Residency Distinction](#)".

About Setting Up Business Profiles for Cache Residency Distinction

You can use the business profiles listed in [Table 16–2](#), with the **CacheResidencyNameValue** entry in an object's validation template, to designate into which memory cache the related object instances are loaded when they are created or changed:

Table 16–2 Business Profiles for Cache Residency Distinction

Business Profile	CacheResidency	Description
Convergent	DEFAULT (0)	Object instances are loaded into both IMDB Cache and batch pipeline memory. These objects are used for rating events in both real time and batch and contain both prepaid and postpaid subscriber information.
Prepaid	REALTIME (1)	Object instances are loaded into IMDB Cache only. These objects are used for rating events in real time and contain prepaid subscriber information.
Postpaid	BATCH (2)	Object instances are loaded into batch pipeline memory only. These objects are used for rating events in batches and contain postpaid subscriber information.
Nonusage	DBONLY (3)	Object instances reside in the BRM database. These objects do not generate any usage and hence do not need to be loaded into Pipeline Manager memory or IMDB Cache.

When an object is created or modified, based on the business profile to which it is associated, BRM reads the validation template to determine the cache residency value and sets the object's PIN_FLD_OBJECT_CACHE_TYPE value accordingly. The object instance is updated in the appropriate memory cache.

Important: The validation templates for a business profile must have compatible cache type values for all objects related to the **/billinfo** object.

Note: The account object does not have a validation template. Account object instances receive their cache type values from their associated **/billinfo** object. All objects outside the **/billinfo** context and associated with the account receive their cache type values from the account. For example, an account-level product has the same PIN_FLD_OBJECT_CACHE_TYPE value as its owner account.

If you do not have cache residency distinction enabled, or do not have a business profile assigned to a reference object, BRM assigns it a Convergent business profile. This sets the cache type to DEFAULT and ensures data is loaded into both memory caches. You can change the default setting by modifying the **/config/business_params** object. See ["Changing the Default Business Profile"](#).

For information on setting up business profiles and validation templates, see ["Managing Business Profiles"](#) in *BRM Managing Customers*.

How BRM Caches Objects in a Convergent Rating System

Before loading data into the cache, BRM determines the PIN_FLD_OBJECT_CACHE_TYPE value of each object and validates that any related objects have valid cache

types. For example, if a **/service/GSM** object has a cache type value of **REALTIME**, any **/purchase_product** objects associated with that service must have a value of **REALTIME** or **DEFAULT**.

During initialization:

- Objects with a **REALTIME** cache value are loaded into IMDB Cache. These include prepaid events.
- Objects with a **BATCH** cache value are loaded into Pipeline Manager memory. These include postpaid events.
- Objects with a **DEFAULT** cache value are loaded into both IMDB Cache and Pipeline Manager memory. These include both prepaid and postpaid events.

During pipeline rating, the following occurs:

1. The **FCT_BillingRecord** module retrieves the **ObjectCacheType** value of a balance group from the **DAT_BalanceBatch** module and publishes it in the EDR. The value can be **BATCH** (postpaid events) or **DEFAULT** (convergent events). See "DAT_BalanceBatch" in *BRM Configuring Pipeline Rating and Discounting*.
2. The **ISC_ObjectCacheTypeOutputSplitter** iScript splits the EDR into two identical files: one that is routed to the **BATCH** stream and one that is routed to the **DEFAULT** stream. See "ISC_ObjectCacheTypeOutputSplitter" in *BRM Configuring Pipeline Rating and Discounting*.
3. Rated Event (RE) Loader loads objects with **BATCH** and **DEFAULT** cache types into the database.
4. RE Loader loads objects with **BATCH** and **DEFAULT** cache types in the database. Then it sends a notification message to IMDB Cache DM about the updates to objects with a **DEFAULT** cache type. IMDB Cache DM retrieves the objects with a **DEFAULT** cache type and updates its cache.

About Changing the Cache Type of an Object

You can change the cache type of an object by associating it with a business profile whose validation template specifies a different cache type value.

Table 16–3 shows what happens when you change an object’s cache type value:

Table 16–3 Cache Type Change Ramifications

Old Cache Type	New Cache Type	How BRM Handles the Change
Prepaid	Postpaid	All subsequent object instances are loaded only into the DAT_AccountBatch and DAT_BalanceBatch modules. The data is loaded into DAT_BalanceBatch immediately; however, the data is loaded into DAT_AccountBatch on the next business event processing for that account or service.
Prepaid	Convergent	All subsequent objects are loaded into the DAT_AccountBatch and DAT_BalanceBatch modules. The data is loaded into DAT_BalanceBatch immediately; however, the data is loaded into DAT_AccountBatch on the next business event processing for that account or service.
Prepaid	Nonusage	No change occurs.

Table 16–3 (Cont.) Cache Type Change Ramifications

Old Cache Type	New Cache Type	How BRM Handles the Change
Postpaid	Prepaid	The object instances previously loaded into the DAT_AccountBatch and DAT_BalanceBatch modules remain in the memory until the next Pipeline Manager initialization.
Postpaid	Convergent	The object instances previously loaded into the DAT_AccountBatch and DAT_BalanceBatch modules are reloaded.
Postpaid	Nonusage	The object instances previously loaded into the DAT_AccountBatch and DAT_BalanceBatch modules are reloaded, and all subsequent objects are not loaded in Pipeline Manager memory from the next initialization of Pipeline Manager.
Convergent	Prepaid	The object instances previously loaded into the DAT_AccountBatch and DAT_BalanceBatch modules remain in the memory until the next Pipeline Manager initialization. The data will not be loaded from the next initialization of Pipeline Manager.
Convergent	Postpaid	All subsequent object instances are reloaded into the DAT_AccountBatch and DAT_BalanceBatch modules.
Convergent	Nonusage	The object instances previously loaded into the DAT_AccountBatch and DAT_BalanceBatch modules are reloaded. The data will not be loaded from the next initialization of Pipeline Manager.
Nonusage	Prepaid	No changes occur.
Nonusage	Postpaid	The object instances are loaded into the DAT_AccountBatch and DAT_BalanceBatch modules. The data is loaded into DAT_BalanceBatch immediately; however, the data is loaded into DAT_AccountBatch on the next business event processing for that account or service.
Nonusage	Convergent	The object instances are loaded into the DAT_AccountBatch and DAT_BalanceBatch module. The data is loaded into DAT_BalanceBatch immediately; however, the data is loaded into DAT_AccountBatch on the next business event processing for that account or service.

Note: Any cache residency value change is not reflected immediately in the DAT_AccountBatch module of Pipeline Manager. The data is loaded only during the subsequent business event processing for that account or service. If you change the cache residency value to a value that is not configured for batch loading, Pipeline Manager does not unload any data. However, the data is not loaded during subsequent Pipeline Manager initialization.

You change an object's cache type by calling the PCM_OP_CUST_CHANGE_BUSINESS_PROFILE opcode. For more information, see "Changing a Bill Unit's Business Profile" in *BRM Managing Customers*.

To configure BRM to handle all objects as postpaid, do not call PCM_OP_CUST_CHANGE_BUSINESS_PROFILE for each object. Instead, do the following:

- Disable the cache residency distinction functionality. See ["Enabling Cache Residency Distinction"](#), but in step 3, set the **CacheResidencyDistinction** value to **disabled**.
- In the **wireless.reg** registry file, set the **ISC_ObjectCacheTypeOutputSplitter iScript's Active** entry to **False**. (See *BRM Configuring Pipeline Rating and Discounting*.)

Configuring BRM for Cache Residency Distinction

The procedures in this section describe how to configure BRM for cache residency distinction.

- Enable cache residency distinction functionality. See ["Enabling Cache Residency Distinction"](#).
- Assign objects a cache residency value. See ["Assigning Objects a Cache Residency Value"](#).
- Set up cache residency validation rules. See ["Setting Up Cache Residency Validation Rules"](#).
- Set the default cache type for the system. See ["Changing the Default Business Profile"](#).
- Override the default business profile. See ["Overriding the Default Business Profile"](#).

Enabling Cache Residency Distinction

By default, cache residency distinction is disabled. You enable it by modifying a field in the **billing** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See ["pin_bus_params"](#) in *BRM Developer's Guide*.

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

2. Search the XML file for following line:

```
<CacheResidencyDistinction>disabled</CacheResidencyDistinction>
```

3. Change **disabled** to **enabled**.

Caution: BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. Changing any parameter in this file affects the associated aspects of the BRM billing configuration.

4. Save and close the file.
5. Use the following command to load the change into the **/config/business_params** object:

```
pin_bus_params bus_params_billing.xml
```

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

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

For more information, see the following topics in *BRM Developer's Guide*:

- Using testnap
 - Reading Objects by Using Object Browser
7. Stop and restart the Connection Manager (CM). For more information, see ["Starting and Stopping the BRM System"](#).
 8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Assigning Objects a Cache Residency Value

To assign a cache residency value to an object, associate it with a prepaid, postpaid, or convergent business profile and define cache validation rules.

For information on setting up business profiles, see "Setting Up Business Profiles and Validation Templates" in *BRM Managing Customers*.

For information on cache residency validation rules, see ["Setting Up Cache Residency Validation Rules"](#).

Setting Up Cache Residency Validation Rules

You define the PIN_FLD_OBJECT_CACHE_TYPE field value by associating an object with a prepaid, postpaid, or convergent business profile and setting the **CacheResidency** value in an object's validation template as shown in [Table 16–4](#).

Table 16–4 Setting Up Cache Residency Validation Rules

Business Profile	CacheResidency Key	CACHE_TYPE Field Value	Memory Cache
Convergent	DEFAULT	0	BRM database, DAT_AccountBatch, DAT_BalanceBatch
Prepaid	REALTIME	1	BRM database only
Postpaid	BATCH	2	DAT_AccountBatch, DAT_BalanceBatch
Nonusage	DBONLY	3	None
Convergent	DEFAULT_INACTIVE	4	BRM database, DAT_AccountBatch, DAT_BalanceBatch
Prepaid	REALTIME_INACTIVE	5	BRM database only
Postpaid	BATCH_INACTIVE	6	DAT_AccountBatch, DAT_BalanceBatch

When you set up a business profile, each validation template you associate with it must have a compatible cache residency value for all objects related to the **/billinfo** object. For example, if you set up a prepaid business profile with validation templates for the **/billinfo**, **/balance_group**, and **/group/sharing/charges** objects, the cache residency value in each object's validation template must be set to **REALTIME**, **DEFAULT**, **DEFAULT_INACTIVE**, or **REALTIME_INACTIVE**.

Note:

- The **CacheResidency** entry is optional. If it is not specified in the validation template, the **PIN_FLD_OBJECT_CACHE_TYPE** value for the object instance gets set to **DEFAULT**.
 - The **/account** object does not have a validation template; it receives its cache residency value from its related **/billinfo** objects.
 - The **/uniqueness** object does not have a validation template; it receives its cache residency value from its related **/service** object.
-
-

Important: If a sponsor group, a top-up group, or a hierarchical account group with nonpaying bill units contains bill units with different business profile settings, the group owner's bill unit must use a compatible business profile. You must add rules to the **/billinfo** object's validation template for the business profile of the group owner's bill unit.

For specific instructions on how to set up business profiles and validation templates for objects, see "Setting Up Business Profiles and Validation Templates" in *BRM Managing Customers*.

Changing the Default Business Profile

By default, all reference object instances get assigned an object cache type value of **DEFAULT** (Convergent business profile) if they have not been associated with a business profile.

To assign a **REALTIME** or **BATCH** object cache type to object instances when they are made, change the default business profile for BRM. This value is defined in the **billing** instance of the **/config/business_params** object.

Note: Before you change the default business profile, define and load the business files into the database. See "Setting Up Business Profiles and Validation Templates" in *BRM Managing Customers*.

Important: If the business profile you set as the default business profile has not been loaded into the database or is not valid, the Convergent business profile is used.

Modify this object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

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

2. Search the XML file for following line:

```
<DefaultBusinessProfile>Convergent</DefaultBusinessProfile>
```

3. Change **Convergent** to one of the following:

- **Prepaid:** Sets the default cache type of all new objects to REALTIME (1).
- **Postpaid:** Sets the default cache type of all new objects to BATCH (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 billing configuration.

4. Save and close the file.
5. Use the following command to load the change into the **/config/business_params** object:

```
pin_bus_params bus_params_billing.xml
```

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

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

For more information, see the following topics in *BRM Developer's Guide*:

- Using testnap
- Reading Objects by Using Object Browser

7. Stop and restart the Connection Manager (CM). For more information, see ["Starting and Stopping the BRM System"](#).
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

Overriding the Default Business Profile

To override the default business profile that is assigned to an object during account creation, pass the business profile POID in the input flist of the PCM_OP_CUST_COMMIT_CUSTOMER opcode.

About Selective Account Loading

Pipeline Manager loads the subscriber data based on the service types configured for batch rating. If the service type is the same for both the prepaid and the postpaid subscribers, Pipeline Manager loads the prepaid subscriber data also.

You can configure your BRM system to load subscriber data selectively in Pipeline Manager based on the business profiles assigned to the accounts. For example, if you use selective account loading, you can load only data for postpaid services instead of postpaid and prepaid data, even though the service type is the same. You can configure any cache residency type data to be loaded into Pipeline Manager memory.

Selective account loading in Pipeline Manager provides:

- Reduced load time during initialization because less data is retrieved from the database.
- Improved memory usage because only selective subscriber information is stored in memory.

When rating the CDRs, Pipeline Manager treats the data as valid only if the cache residency value of the data at the time of the event matches with the values configured for loading data into Pipeline Manager.

Configuring Pipeline Manager for Selective Account Loading

You can configure Pipeline Manager to load selective accounts during initialization by enabling selective account loading functionality. See ["Enabling Selective Account Loading"](#).

Note: For selective account loading functionality, you must enable the cache residency distinction parameter. See ["Enabling Cache Residency Distinction"](#).

Enabling Selective Account Loading

By default, selective account loading functionality is disabled. You can enable this functionality by loading and configuring an optional business parameter, **CacheResidenciesForBatchPipeline**, in the *BRM_home/sys/data/config/bus_params_selective_loading.xml* file.

To load and configure the values in the **CacheResidenciesForBatchPipeline** business parameter:

1. Search the **bus_params_selective_loading.xml** file for following line:


```
<CacheResidenciesForBatchPipeline>0,1</CacheResidenciesForBatchPipeline>
```
2. Change **0,1** to any cache residency values of accounts you want to load, separated by comma. For example, to load convergent, prepaid, and postpaid accounts into Pipeline Manager, change **0,1** to **0,1,2**.
3. Save and close the file.
4. Use the following command to load the change into the **/config/business_params** object:

```
pin_bus_params bus_params_selective_loading.xml
```

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

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

For more information, see the following topics in *BRM Developer's Guide*:

- Using testnap
 - Reading Objects by Using Object Browser
6. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).
 7. Stop and restart Pipeline Manager.
 8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see ["pin_multidb"](#).

The following is a sample **bus_params_selective_loading.xml** file:

```
BusParamConfigurationClass>
  <BusParamsSelectiveLoading>
    <CacheResidenciesForBatchPipeline>0,1,2</
      CacheResidenciesForBatchPipeline >
  </BusParamsSelectiveLoading>
</BusParamConfigurationClass>
```

Here, **0,1,2** specifies the cache residency types DEFAULT, REALTIME, and POSTPAID. After the **CacheResidenciesForBatchPipeline** business parameter is loaded, Pipeline Manager loads all accounts with Convergent, Prepaid, and Postpaid business profiles.

Configuring Pipeline Manager to Process Prepaid CDRs

By default, Pipeline Manager rates only one event type per service (for example, delayed session event for GSM telephony service).

If the selective account loading functionality is enabled, you can load prepaid subscribers in Pipeline Manager. However, Pipeline Manager rejects any prepaid CDRs of the prepaid subscribers if the delayed event type configured for batch rating is not present in any of the products owned by the service or account. This is because of the difference in the prepaid and postpaid event types. For example, real-time session event for prepaid events and delayed session event for postpaid events.

To allow the CustomerSearch module to accept the prepaid CDRs, you can use the FCT_Account module **DisableRatingProductCheck** registry entry to configure how product rating is checked:

- If you enable this entry, FCT_Account does not reject any prepaid CDRs of the prepaid subscribers if the configured event for batch rating is not present in any of the products owned by the service or account. Pipeline Manager does not rate CDRs, but the DAT_AccountBatch plug-in provides the subscriber information. You can use this subscriber information for any customized processing. For example, to pass rated roaming prepaid CDRs through Pipeline Manager, you can customize the action on the CDRs based on the subscriber information.
- If you disable this entry, the FCT_Account rejects any prepaid CDRs of the prepaid subscribers if the configured event for batch rating is not present in any of the products owned by the service or account. By default, **DisableRatingProductCheck** is set to **False**.

Customizing Cache Residency Distinction

You can customize cache residency by performing any of the following tasks:

- Configure nonreference storable classes or custom storable classes for cache residency distinction. See ["Configuring Nonreference Objects for Cache Residency Distinction"](#).

- Create, change, or delete business profiles. See "Setting Up Business Profiles and Validation Templates" in *BRM Managing Customers*.
- Create new validation templates (subclasses of the `/config/template` base class). See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.
- Customize the `ISC_ObjectCacheTypeOutputSplitter` iScript. See "ISC_ObjectCacheTypeOutputSplitter" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Nonreference Objects for Cache Residency Distinction

For specific instructions on how to create and modify storable classes, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

To configure nonreference objects:

1. Use Storable Class Editor to define a new class or modify an existing BRM class, and add the `RESIDENCY_TYPE` attribute and `PIN_FLD_OBJECT_CACHE_TYPE` field to the class. See the Storable Class Editor Help.
2. Create a validation template for the new storable class. See "Setting Up Business Profiles and Validation Templates" in *BRM Managing Customers*.

Important: This is necessary to validate the cache residency value. Be certain to set the **CacheResidency** key value to **Prepaid**, **Postpaid**, or **Convergent**.

3. Associate the new validation template with a business profile.
4. Add or modify iScript validation rules to determine the `PIN_FLD_OBJECT_CACHE_TYPE` value for related object instances.

Note: BRM does not supply iScript rules for business profile validation. Instead, you must create your own rules. For information about creating iScript validation rules, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

5. Configure the iScript in the validation templates to return a list of object POIDs and the cache residency value for each one.
6. Customize the appropriate opcode to call the validation template. Perform a `WRITE_FLDS` operation on the object instances to set their `PIN_FLD_OBJECT_CACHE_TYPE` value.

Part IV

Troubleshooting BRM

Part IV describes how to troubleshoot an Oracle Communications Business and Revenue Management (BRM) system. It contains the following chapters:

- [Resolving Problems in Your BRM System](#)
- [Reference Guide to BRM Error Codes](#)
- [Pipeline Manager Error Messages](#)

Resolving Problems in Your BRM System

This chapter provides guidelines to help you troubleshoot problems with your Oracle Communications Billing and Revenue Management (BRM) system. You can find information about interpreting error messages, diagnosing common problems, and contacting Oracle Technical Support.

Before you read this chapter, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

For information on problems related to poor performance, see ["Improving BRM Performance"](#).

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 Customer Center 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 to Connect and Configure Components](#)
- [Starting and Stopping the BRM System](#)

If you still cannot resolve the problem, contact Oracle Technical Support as described in ["Getting Help with BRM Problems"](#).

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 must be managed and 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.

Finding Error Log Files

BRM records system activity in a set of log files, one for each component or application. If a component of your BRM system has a problem, you can find the error message in the log file for that component. For information on locating a log file, see ["Using Logs to Monitor Components"](#).

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. See ["Setting the Reporting Level for Logging Messages"](#).

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 to correlate the log messages from all BRM components related to a single client operation. This information can be used to sort error messages and to identify the set of error messages generated from a single operation. See ["Logging External User Information in Error Logs"](#).

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 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
2:CT1255:Account_
Manager:1948:1684:63:1063403309:14:root.0.0.0.1::user1:123456789
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

```
2:CT1255:Account_
Manager:1948:1684:63:1063403309:14:root.0.0.0.1::user1:123456789
```

- 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

```
2:CT1255:Account_
Manager:1948:1684:63:1063403309:14:root.0.0.0.1::user1:123456789
E Fri Sep 12 14:50:05 2003 db2.corp dm:12250 dm_subr.c(1.58):351
2:CT1255:Account_
Manager:1948:1684:63:1063403309:14:root.0.0.0.1::user1:123456789
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 deduce 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: Problem with Customer Center

When you try to add a new account in Customer Center, you see the following message:

A problem occurred while attempting to create an account. The error is likely related to some invalid field, missing required information, or duplicate information. Please check fields carefully, and try again.

- Check the **cm.pinlog** file, which shows the following error message:

```
D Mon Apr 18 12:41:17 2003 turnip cm:16798 fm_bill_purchase.c:493
2:CT1255:Account_
Manager:1992:1940:63:1050694800:5:root.0.0.0.1::user1:123456789
qnty (200.000000) > max (1.000000)...
E Mon Apr 18 12:41:17 2003 turnip cm:16798 fm_bill_purchase.c:196
2:CT1255:Account_
Manager:1992:1940:63:1050694800:5:root.0.0.0.1::user1:123456789
op_bill_purchase error [location=<PIN_ERRLOC_FM:5>
class=<PIN_ERRCLASS_APPLICATION:4> errno=<PIN_ERR_BAD_VALUE:46>
field num=<PIN_FLD_PURCHASE_MAX:4,237> recid=<0> reserved=<0>]
```

The debugging message shows that the account creation routine is trying to purchase a product. The error indicates that the purchase quantity (200) is more than the allowed quantity of the product (1). Because the account creation processes use deals to purchase products, this error probably means that the deal has been defined incorrectly. Look in the price list to check the maximum allowable purchase amount for the product.

Example 3: 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
2:CT1255:dm:28492:1:0:1063265316:0:root.0.0.0.1::user1:123456789
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 4: 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
2:CT1255:dm:28509:1:0:1061195411:7:root.0.0.0.1::user1:123456789
ORACLE error: do_sql_insert: obndrv: code 1036, op 28, peo 0
=ORA-01036: illegal variable name/number
was binding ":poid_DB" buf 0x195b180, buf1 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. For information on how to gather the SQL statements generated by the Oracle database or DM, see ["Increasing the Level of Reporting for a DM"](#).

Diagnosing Some Common Problems with BRM

This section lists some common problems with BRM and shows you how to diagnose the error messages and resolve the problems.

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 master Oracle DM 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*):

```
- 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: 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

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

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. See ["Starting and Stopping the BRM System"](#).

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 HP-UX IA64: To see how many sockets are available:

```
netstat -n -f inet | grep '^tcp' | wc -l
```

On Linux: To see how many sockets are available:

```
netstat -n -A inet -t | wc -l
```

On AIX: To see how many sockets are available:

```
netstat -n -f inet | grep tcp
```

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.

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 hang 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.

Important: 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 'cmlddl [application]'
```

On HP-UX IA64:

```
ps -eo pid,vsz,flags,state,sz,pmem,comm | egrep 'cmlddl [application]'
```

On Linux:

```
ps -eo pid,vsz,f,s,sz,pmem,comm | egrep 'cmlddl [application]'
```

On AIX:

```
ps -eo pid,vsz,dpgsz,THREAD,comm | egrep 'cmlddl [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 or RADIUS Manager operations, 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. See ["Starting and Stopping the BRM System"](#).

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. See ["Starting and Stopping the BRM System"](#).

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.

To check for and clean up unused memory on AIX:

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 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. See ["Starting and Stopping the BRM System"](#).

Problem: Java Out of Memory Error

When using GUI applications such as Pricing Center, Suspense Manager, and Customer Center or batch applications such as Invoice formatter, you may 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_bigsizes** and **dm_shmsizes** 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 master 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 "Disabling the Welcome Message" 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`.

Getting Help with BRM Problems

If you cannot resolve your problems with BRM, contact Oracle Technical Support.

Before You Contact Technical Support

Problems can often be fixed simply by shutting down BRM and restarting the computer that the BRM system runs on. See ["Starting and Stopping the BRM System"](#).

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.

Reporting Problems

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](#)".
 - You can collect pipeline information by using the **piperev** script. For information, see "[Checking Pipeline Component Version Numbers](#)".
-
-

When you are ready, report the problem to Oracle Technical Support.

Reference Guide to BRM Error Codes

This chapter lists the Oracle Communications Billing and Revenue Management (BRM) error locations, classes, and 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 18–1](#) lists the BRM error locations.

Table 18–1 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.
PIN_ERRLOC_CM	3	Connection Manager. Common causes include an unknown opcode or an input flist missing the required Portal object ID (POID) field.

Table 18–1 (Cont.) BRM Error Locations

Error Location	No.	Source of the Error
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_NMGR	12	An error occurred within Node Manager.
PIN_ERRLOC_INFMGR	13	An error occurred within System Manager.
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.
PIN_ERRLOC_RTP	19	An error occurred within a real-time pipeline.
PIN_ERRLOC_ADT	21	An error occurred within the ADT lib (iScript).

BRM Error Classes

[Table 18–2](#) lists the BRM error classes.

Table 18–2 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.
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. A small window exists during the commit where a network failure can leave the system unsure of whether the commit occurred. Hence, the application must 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 in 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 18–3 lists the BRM error codes.

Table 18–3 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 could not 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 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_NOT_FOUND	3	BRM could not find a value. This error code does not 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 could not read from the network or some other IO device, probably because the connection was cut off unexpectedly.
PIN_ERR_NO_SOCKET	8	BRM could not 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 does not 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 a storable object because the requested ID is already used by another object.
PIN_COMPARE_EQUAL	11	This code does not indicate an error. The billing FM uses this code for internal operations.
PIN_COMPARE_NOT_EQUAL	12	This code does not 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 does not 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 it 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.
PIN_ERR_DUP_SUBSTRUCT	17	Information not available.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
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 storable 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 could not 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 does not work, report the problem to the OS vendor.
PIN_ERR_DM_ADDRESS_LOOKUP_FAILED	25	The Connection Manager could not 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 could not 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 could not 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 could not assign a storable object ID.
PIN_ERR_BAD_CREATE	34	A routine could not create an object.
PIN_ERR_BAD_FIELD_NAME	35	Mapping error from field home to type.
PIN_ERR_BAD_OPCODE	36	Undefined opcode used.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
PIN_ERR_TRANS_ALREADY_OPEN	37	An application attempted to open a transaction when one was already open on the same storable 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 stop 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 could not 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 storable object does not 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 could not interpret data from the database. The data is not 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 is not 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.
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.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
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 could not create an object because one or more values were invalid or fields do not allow updating.
PIN_ERR_BAD_LOGIN_RESULT	56	The application could not connect to the CM. Check the login name and password.
PIN_ERR_CM_ADDRESS_LOOKUP_FAILED	57	The application could not 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 redirection 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 did not find an expected STEP_NEXT or STEP_END .
PIN_ERR_STORAGE_DISCONNECT	61	BRM lost the connection with the database during the middle of a transaction and could not reestablish the connection. See "Configuring Oracle Databases" in <i>BRM Installation Guide</i> .
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 could not start the process.
PIN_ERR_STOP_FAILED	68	BRM could not stop the process.
PIN_ERR_INVALID_QUEUE	69	No information available.
PIN_ERR_TOO_BIG	70	No information available.
PIN_ERR_BAD_LOCALE	71	BRM does not 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.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
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 is 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 does not support canonicalization for the locale of the client application.
PIN_ERR_CURRENCY_MISMATCH	78	A subordinate bill unit or sponsored account has a different account currency than the parent or sponsor 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 stops 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 initialize cm_cache with zero size.
PIN_ERR_INVALID_OBJECT	94	The POID is invalid. This error occurs when an object is accessed whose <i>database_number</i> field of the POID is NULL.
PIN_ERR_VALIDATE_ADJUSTMENT	95	The validate adjustment policy opcode fails.
PIN_ERR_SYSTEM_ERROR	96	A generic system error occurred while the application was running.
PIN_ERR_BILLING_ERROR	97	An error occurred during the billing run.
PIN_ERR_AUDIT_COMMIT_FAILED	98	Commit for the audit tables failed.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
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	Deals or plans 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	An Oracle 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_FEATURE_DISABLED	110	The feature is disabled.
PIN_ERR_INVALID_STATE	111	The state is invalid.
PIN_ERR_NO_SECONDARY	112	No secondary instance exists.
PIN_ERR_ACCT_CLOSED	113	The account is in the Closed state.
PIN_ERR_EM_CONNECT_FAILED	114	The connection between the CM and EM failed.
PIN_ERR_FAST_FAIL	115	A fast fail occurred in dual timeout.
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.

Table 18–3 (Cont.) BRM Error Codes

Error Code	No.	Description
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.
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.

Pipeline Manager Error Messages

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager error messages.

Note:

- Many error descriptions include the string *value*. This string is replaced with the appropriate value by the module logging the error.
 - Modules that are not listed in this chapter do not log module-specific error messages. However, modules can return pipeline framework error messages. For information on framework error messages, see ["Pipeline Framework Error Messages"](#).
-

Pipeline Framework Error Messages

[Table 19–1](#) lists the Pipeline Framework error messages.

Table 19–1 Pipeline Framework Error Messages

Error Message	Description
ERR_A_CUSTOMER_NOT_FOUND	A-Customer not found (<i>value</i>).
ERR_ACTIVATED_DATE_INVALID	Contract <i>value</i> has an invalid activation date.
ERR_ADD_DATABLOCK	Cannot add Datablock ' <i>value</i> ' to EDR-C.
ERR_ALIAS_IS_IN_WRONG_BLOCK	The alias <i>value</i> is in the wrong block.
ERR_BAD_SCHEMA_SOURCE	Bad schema source: <i>value</i> .
ERR_BLOCK_DESC_NOT_FOUND	Block description not found (<i>value</i>).
ERR_BLOCKID_UNKNOWN	Can't find an index for id <i>value</i> .
ERR_BUILD_DESC_TREE	Error while opening/reading description tree (<i>value</i>).
ERR_CALENDAR_PLUGIN_INV	Calendar data module invalid.
ERR_CALLTYPE_INVALID	Invalid call type: <i>value</i> : <i>value</i> .
ERR_CALLTYPE_NOT_FOUND	No call type found for EDR (<i>value</i>).
ERR_CAN_NOT_GET_FACTORY	Cannot get factory ' <i>value</i> '.
ERR_CANCEL_FAILED_INPUT	Cancel failed in input-controller.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_CANCEL_TRANSACTION	Module ' <i>value</i> ' failed to cancel transaction ' <i>value</i> '.
ERR_CANNOT_DECRYPT_PASSWORD	Cannot decrypt password ' <i>value</i> '; user ' <i>value</i> '.
ERR_CANNOT_FIND_EVENT_HANDLER_PROC	Cannot locate the event handler daemon!
ERR_CANNOT_FORK	Cannot create child process.
ERR_CANNOT_GET_DMT_DMCONNECTION	<i>value</i> : Cannot get a DMT::DMConnection
ERR_CANNOT_INIT_DB_VERSION	Cannot initialize database version, error ' <i>value</i> '.
ERR_CANNOT_INIT_INPUT_STREAM	Cannot initialize the input stream object.
ERR_CANNOT_INIT_INPUT_STREAM_INTERFACE	Cannot initialize the input stream interface object.
ERR_CANNOT_INIT_OUTPUT_COLLECTION	Cannot initialize the output collection.
ERR_CANNOT_INIT_OUTPUT_MODULE	Cannot initialize the output module.
ERR_CANNOT_INIT_OUTPUT_STREAM	Cannot initialize the output stream object.
ERR_CANNOT_INIT_OUTPUT_STREAM_INTERFACE	Cannot initialize the output stream interface object.
ERR_CANNOT_JOIN_EVENT_HANDLER_PROC	Cannot connect to event handler process: <i>value</i>
ERR_CANNOT_OPEN_DATABASE	Cannot open database ' <i>value</i> '; user ' <i>value</i> '; password ' <i>value</i> '; server message ' <i>value</i> '.
ERR_CANNOT_RENAME_OUTPUT_FILE	Cannot rename temporary output file ' <i>value</i> '.
ERR_CHARGE_ITEM_INVALID	ChargeItem <i>value</i> invalid.
ERR_CHARGED_ZONE_NOT_FOUND	The EDR charged zone cannot be found for ' <i>value</i> '.
ERR_CIBER_RET	CIBER return: retReason <i>value</i> , retCode <i>value</i> , fieldID <i>value</i> , ruleID <i>value</i> .
ERR_CLIMAP_FILENAME_EMPTY	Empty cli mapping file name specified.
ERR_COMMIT_TRANSACTION	Module ' <i>value</i> ' failed to commit transaction ' <i>value</i> '.
ERR_CON_ATTACHED_FIELD	The attached field information has not the right format. Must be BLOCKNAME.FIELDNAME, is <i>value</i>).
ERR_CONTROLLER_CONFIGURATION	Pipeline controller configuration has error in ' <i>value</i> '.
ERR_CONTROLLER_HAS_WRONG_TYPE	Pipeline controller has wrong type in ' <i>value</i> '.
ERR_CONVERSION_BAS_DATE	<i>value</i> could not be converted to BAS_Date.
ERR_CONVERSION_FAILED	EDR conversion failed (<i>value</i>).
ERR_CONVERSION_INT	<i>value</i> is no valid integer value.
ERR_CORBA_EXCEPTION	CORBA exception: <i>value</i> .
ERR_CREATE_ALIAS_MAP_INDEX	No AliasMap entry found for Reference ' <i>value</i> ' and logical Name ' <i>value</i> '.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_CREATE_EDR_INDEX	EDR index creation failed: <i>value</i> (name=` <i>value</i> `, key=` <i>value</i> ` and reference=` <i>value</i> `).
ERR_CREATE_INDEX	EDR index creation failed: <i>value</i>
ERR_CREATE_INPUT_PARSER	Failed to create input parser: <i>value</i>
ERR_CREATE_INSTANCE	Error creating instance of <i>value</i>
ERR_CREATE_OBJECT_FAILED	Cannot create object ' <i>value</i> ' (invalid (NULL) pointer).
ERR_CREATE_OUTPUT_PARSER	Failed to create output parser: <i>value</i>
ERR_CREATE_SCRIPT	Error loading script <i>value</i> : <i>value</i> .
ERR_CREATE_THREAD_FAILED	Cannot create thread instance for ' <i>value</i> '; invalid thread body.
ERR_CUG_FILENAME_EMPTY	Empty closed user group file name specified.
ERR_CUST_A_IDENTIFICATION_UNKNOWN	Customer identification technique for used service (' <i>value</i> ') not found
ERR_CUST_A_VALUE_NOT_FOUND	Missing value for field ' <i>value</i> ' of Customer A
ERR_CUST_FILE_VERSION	Illegal customer file version <i>value</i> .
ERR_CUST_FILENAME_EMPTY	Empty customer file name specified.
ERR_CUSTOMER_DATA_INVALID	Invalid customer data.
ERR_DAT_PREFDESC_INS_TREE_DB	Can't insert line <i>value</i> from table <i>value</i> into prefix description table.
ERR_DAT_PREFDESC_INS_TREE_FILE	Can't insert line <i>value</i> from file <i>value</i> into prefix description table.
ERR_DATA_INVALID	The data in field <i>value</i> is invalid.
ERR_DATA_PLUGIN_INVALID	Module ' <i>value</i> ' is invalid.
ERR_DATA_PLUGIN_NOT_FOUND	Module ' <i>value</i> ' cannot be found in the DataPool.
ERR_DATABASE	Database error ' <i>value</i> '.
ERR_DB_COMMIT_TRANSACTION	Cannot commit database transaction ' <i>value</i> '.
ERR_DB_CONNECTION_MODULE	Database connection module is invalid.
ERR_DB_CONNECTION_NOT_VALID	Could not connect to database.
ERR_DB_NUMBER_OF_ROWS	Statement ' <i>value</i> ' does not return exactly one row.
ERR_DB_START_TRANSACTION	Error starting database transaction: ' <i>value</i> '
ERR_DB_STATEMENT_EXECUTE	Cannot execute database statement ' <i>value</i> ', message ' <i>value</i> '.
ERR_DB_VERSION_CHECK	Wrong database version. Check module and database version.
ERR_DB_VERSIONS_NOT_FOUND	Database versions ' <i>value</i> ' not found.
ERR_DD_NOT_READ	Cannot read the data dictionary.
ERR_DEF_IS_INCOMPLETE	The field definition ' <i>value</i> ' is incomplete.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_DEFAULT_BLOCK_NOT_FOUND	The specified default block name <i>value</i> does not exist in the description.
ERR_DEFAULT_WITH_WRONG_ID	Output Stream <i>value</i> : The default block has a wrong id. Check your format description.
ERR_DELETE_FILE	Cannot delete file ' <i>value</i> '.
ERR_DELETE_OUTPUT_FILE	' <i>value</i> ': Cannot delete output file ' <i>value</i> '.
ERR_DIR_EMPTY	Reading from empty directory ' <i>value</i> '.
ERR_DIR_NOT_ACCESSIBLE	Directory ' <i>value</i> ' is not accessible.
ERR_DIR_NOT_WRITEABLE	Directory <i>value</i> is not writable.
ERR_DLOPEN_FAILED	Cannot open shared library ' <i>value</i> '; <i>value</i> . Make sure the LD_LIBRARY_PATH_64 environment variable includes <i>Pipeline_home/lib</i> .
ERR_DLSYM_FAILED	Cannot get address of generator function ' <i>value</i> '; <i>value</i> .
ERR_DONE_PATH_NOT_FOUND	Entry for done path not found in registry.
ERR_DOUBLE_ALIAS_NAME	The reference to the alias <i>value</i> exist more than one times.
ERR_DOUBLE_SEQ_NUMBER	Double sequence number found (sequence number: ' <i>value</i> ').
ERR_DOUBLE_TRANS_MODULE	Transaction module ' <i>value</i> ' was attached more than once.
ERR_DOUBLE_TRANSACTION_ID	Double transaction id ' <i>value</i> ' found.
ERR_DUPLICATE_IRULE_PARAMETER	Duplicate iRule parameter ' <i>value</i> ' found in file ' <i>value</i> '.
ERR_DUPLICATE_NUMPORTDATA	<i>value</i> : Duplicate number portability data found for the CLI <i>value</i> and the Portation TimeStamp <i>value</i>
ERR_EDR_ALIAS_NOT_FOUND	The specified field alias <i>value</i> couldn't founded.
ERR_EDR_BLOCK_NOT_FOUND	The specified block alias <i>value</i> couldn't founded.
ERR_EDR_BUILD_RECORD_NOT_FILLED	' <i>value</i> ' - EDR buildt record field not filled.
ERR_EDR_CREATE	Failed to create new EDR container.
ERR_EDR_FACTORY_NOT_FOUND	EDR-Factory ' <i>value</i> ' not found.
ERR_EDRTRACE_STREAMLOG_CREATION_FAIL	Error in EDR trace stream log creation.
ERR_EMPTY_CHARGEPACKET_LIST	No charge-packets found in charge breakdown record.
ERR_ERROR_PATH_NOT_FOUND	Entry for error path not found in registry.
ERR_ERROR_RATE_ALREADY_DEFINED	Error rate for ' <i>value</i> ' already specified.
ERR_ERROR_RATE_VALUE_NOT_SPECIFIED	No value specified for error ' <i>value</i> '.
ERR_EVAL_ENVIRONMENT	Cannot evaluate environment ' <i>value</i> '.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_FAILURE_FSM	Failure in finite state machine: <i>value</i> .
ERR_FILE_CLOSE_OS	<i>value</i> : Cannot close file ' <i>value</i> '.
ERR_FILE_EOF	Tried to read past end of file ' <i>value</i> '.
ERR_FILE_EXIST	File ' <i>value</i> ' exist.
ERR_FILE_MOVE_OS	<i>value</i> : Cannot move file ' <i>value</i> ' to ' <i>value</i> '.
ERR_FILE_NOT_FOUND	File ' <i>value</i> ' not found.
ERR_FILE_NOT_MOVED	File ' <i>value</i> ' could not be moved to ' <i>value</i> '.
ERR_FILE_NOT_WRITABLE	File ' <i>value</i> ' is not writable.
ERR_FILE_OPEN_OS	<i>value</i> : Cannot open file ' <i>value</i> '.
ERR_FILE_READ_ERR	Error reading from file ' <i>value</i> '.
ERR_FILE_READ_OS	<i>value</i> : Error reading from file ' <i>value</i> '.
ERR_FILE_REMOVE_OS	<i>value</i> : Cannot remove file ' <i>value</i> '.
ERR_FILE_WRITE_ERR	Error writing into file ' <i>value</i> '.
ERR_FILE_WRITE_OS	<i>value</i> : Error writing into file ' <i>value</i> '.
ERR_FILENAME_MISSING	File name not set for ' <i>value</i> '.
ERR_FLIST_INPUT_ERROR	Error while processing FLIST message: <i>value</i>
ERR_GAP_IN_SEQ_NUMBER	Gap in sequence number found (sequence number: ' <i>value</i> ').
ERR_GETTING_DATADESCR	Failed to get the data description.
ERR_GRAMMAR_SYMBOL_LOOKUP	Symbol lookup for ' <i>value</i> ' failed: <i>value</i>
ERR_ILL_RECORD_TYPE	Illegal record type ' <i>value</i> ' found.
ERR_ILLEGAL_STREAM_NUM	Tried to use illegal stream number ' <i>value</i> ' for output.
ERR_IN_RECEIVED_MESSAGE	Message <i>value</i> was invalid.
ERR_IN_SECTION	Error in section <i>value</i> .
ERR_INCORRECT_FILLER_LENGTH	Invalid record Filler length, expected: ' <i>value</i> ', received: ' <i>value</i> '.
ERR_INCORRECT_FORMAT_OBJ	The format description object couldn't be founded or is invalid.
ERR_INDEX_NOT_CREATED	Couldn't create the index for alias <i>value</i> .
ERR_INDEX_NOT_FOUND	Container index not found.
ERR_INIT_EDR_ITERATOR	Failed to initialize EDR iterator for ' <i>value</i> '.
ERR_INIT_SEG_TARIFF_LINK	Failure during initialization of tariff segment link table.
ERR_INIT_TSC_MAPTABLE	Failed to init map table: <i>value</i> .
ERR_INIT_XERCES	Error: Xerces-c Initialization. Exception message: <i>value</i>
ERR_INPUT_DONE_FILE_NOT_MOVED_TO_ERR	' <i>value</i> ': Cannot move done file ' <i>value</i> ' to error file ' <i>value</i> '.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_INPUT_DONE_FILE_NOT_MOVED_TO_INPUT	'value': Cannot move done file 'value' to input file 'value'.
ERR_INPUT_FILE_NOT_MOVED	'value': Cannot move input file 'value' to temporary file 'value'.
ERR_INPUT_MAPPING_FAILED	Input mapping `value' failed: value.
ERR_INPUT_PATH_NOT_FOUND	Entry for input path not found in registry.
ERR_INPUT_REQUEST_ROLLBACK	The input has requested a rollback (reason=value).
ERR_INPUT_TEMP_FILE_NOT_MOVED	'value': Cannot move temporary input file 'value' to input file 'value'.
ERR_INPUT_TEMP_FILE_NOT_MOVED_TO_DONE_ERR	'value': Cannot move temporary file 'value' to done or err file 'value'.
ERR_INSERT_HASH	Failure during insert in hash map.
ERR_INSERTING_CLI	Error loading cli 'value' (probably duplicated)
ERR_INSUFFICIENT_MEMORY	Insufficient memory available.
ERR_INVALID_DATABASE_VALUE	Database value for field 'value' is invalid.
ERR_INVALID_DATE	Can't build date 'value' for cli 'value'.
ERR_INVALID_DATETIME	value. Cannot build datetime 'value' for cli 'value'.
ERR_INVALID_FCI_COLL_ENTRIES	Invalid number of FCI collection entries (value).
ERR_INVALID_FCI_COLL_ORDER	Invalid order of FCI collection entries (value). This error occurs when buffers are configured in multiple function pools. In this configuration, each buffer must have a unique name.
ERR_INVALID_FIRST_CALL_TIMESTAMP	Invalid first call timestamp: value, calculated: value.
ERR_INVALID_HA_ROLE	The peer instance has already assumed the value role
ERR_INVALID_INPUT_RECORD	Check length, numeric values or date fields for their correctness. (record:value)
ERR_INVALID_LAST_CALL_TIMESTAMP	Invalid last call timestamp: value, calculated: value.
ERR_INVALID_LINE_LENGTH	The input line length for record number value is invalid.
ERR_INVALID_PATTERN	Directory pattern 'value' is invalid.
ERR_INVALID_PLUGIN_STATE	Invalid internal module state in value.
ERR_INVALID_QUEUE_SIZE	Queue size < 0.
ERR_INVALID_RECORD_LENGTH	Defined RecordLength (value) does not match length (value) of read line.
ERR_INVALID_RECORD_NUMBER	Invalid number of records: value, counted: value.
ERR_INVALID_REG_BASE_NAME	Registry base name of 'value' does not match 'value'.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_INVALID_REG_ENTRIES	Invalid Registry Entries. <i>value</i>
ERR_INVALID_REG_VALUE	Invalid value ' <i>value</i> ' for ' <i>value</i> '.
ERR_INVALID_REJECT_STREAM_NUMBER	Stream number is out of range.
ERR_INVALID_SEQ_NUM	Invalid sequence number ' <i>value</i> '.
ERR_INVALID_SEQ_VALUE	The configuration value for <i>value</i> is invalid (<i>value</i>).
ERR_INVALID_SOCIAL_NO	Invalid social number ' <i>value</i> '.
ERR_INVALID_STATE	Received EDR invalid in the current state.
ERR_INVALID_THREAD_STATE	Invalid thread state in ' <i>value</i> '; <i>value</i> ; <i>value</i> .
ERR_INVALID_TOKEN_COUNT	Number of HA role mediator token should be one but found ' <i>value</i> '.
ERR_INVALID_TOKEN_DB_NO	Invalid HA role mediator token database number. Found ' <i>value</i> ' and expected to be ' <i>value</i> '.
ERR_LAST_LOAD_RELOAD_FAILED	The last load/reload operation has failed.
ERR_LEN_IS_MISSING	The first item in field definition <i>value</i> must a number.
ERR_LINE_NOT_IDENTIFIED	The line couldn't identified: <i>value</i>
ERR_LINE_NOT_INSERTED_DOUBLE	Could not insert line into message DB (double key). Line <i>value</i>
ERR_LINE_NOT_INSERTED_INVALID	Could not insert line into message DB (invalid key). Line <i>value</i>
ERR_LINK_TABLE_INVALID	The link table <i>value</i> is invalid.
ERR_LOADING_ABORTED	Loading data aborted after <i>value</i> records.
ERR_LOADING_CUSTOMER_DATA	Loading customer data failed.
ERR_LOADING_DBTABLE	Error while loading database table <i>value</i> .
ERR_LOADING_TIMEMODEL	Loading time model failed ' <i>value</i> '.
ERR_MAPPING_TABLE_INVALID	The mapping table is invalid.
ERR_MBI_INPUT_ERROR	Error while processing MBI message: <i>value</i>
ERR_MEM_MON_MEMORY_LIMIT	<i>value</i> Reached specified memory usage limit.Usage: <i>value</i> KB, available: <i>value</i> KB
ERR_MEM_MON_PROCESS_LIMIT	<i>value</i> Reached process size limit.Size: <i>value</i> KB, limit: <i>value</i> KB
ERR_MISSING_ARGUMENT	Argument ' <i>value</i> ' not in message ' <i>value</i> '.
ERR_MISSING_LOG_FILE_NAME	Log output file name is missing.
ERR_MISSING_MESSAGE_FILE_NAME	Message file name is missing.
ERR_MISSING_REFERENCE_FIELD	Find some container references without a field reference in block <i>value</i> .
ERR_MISSING_REFERENCE_NAME	Missing reference name in block description.
ERR_MISSING_VALUES_FOR_FIELD	Find a reference field entry without id's in block <i>value</i> .

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_MODULE_NOT_EXIST	The module ' <i>value</i> ' which was configured as an event originator does not exist.
ERR_MULTIPLE_RESTART_FILES	Found more than one restart file in directory ' <i>value</i> '.
ERR_NO_CLI	No cli in input record.
ERR_NO_CUSTOMER	No customer data for cli <i>value</i> in input record.
ERR_NO_CUSTOMER_DATA	No customer data present.
ERR_NO_CUSTOMER_PLUGIN	No customer plug-in present.
ERR_NO_DATABASE_PLUGIN	No database plug-in present.
ERR_NO_DEFAULT_OUTPUT_DEVICE	No default output device.
ERR_NO_DEFAULT_SENTENCE	There is no default sentence defined in the format description.
ERR_NO_DIR	Directory ' <i>value</i> ' not accessible.
ERR_NO_EDRFACTORY	Can't get the factory to create EDRs in <i>value</i> .
ERR_NO_EVENTHANDLER_FOUND	Event handler not found in module ' <i>value</i> '.
ERR_NO_INDEX	Index ' <i>value</i> ' not found.
ERR_NO_MESSAGE_FILE	There are no message file found. Path : <i>value</i>
ERR_NO_ORIGINAL_RECORD	Missing the original block.
ERR_NO_PATH_NAME	No path name given.
ERR_NO_REQUEST	Request <i>value</i> returned with no value.
ERR_NO_SEQ_VALUE	Sequence field " <i>value</i> " in sequence control file has no value.
ERR_NO_SPLITTING_PERFORMED	No splitting performed (spec-sys = <i>value</i>).
ERR_NO_SUBSCRIBER	No subscriber data for cli <i>value</i> in input record.
ERR_NOSP_ID_NOT_FOUND	NOSP-Id not found for Frm= <i>value</i> and AreaCode= <i>value</i> .
ERR_NOT_USABLE	The object ' <i>value</i> ' is not usable.
ERR_NOT_USABLE_REASON	Module is not usable: <i>value</i> .
ERR_NUMBER_OF_FIELDS_IN_RECORD	Found ' <i>value</i> ' instead of ' <i>value</i> ' fields in record ' <i>value</i> ' <i>value</i> .
ERR_OBJ_ALREADY_REGISTERED	' <i>value</i> ' is already registered as ' <i>value</i> '.
ERR_OBJ_NOT_FOUND	The object ' <i>value</i> ' could not be found.
ERR_OBJ_NOT_INITIALIZED	The object ' <i>value</i> ' is not initialized.
ERR_OBJ_NOT_REGISTERABLE	The object ' <i>value</i> ' could not be registered.
ERR_OBJ_NOT_REGISTERED	The object ' <i>value</i> ' is not registered.
ERR_OFF_MIN_GREATER_MAX	The min offset is greater than the max offset.
ERR_ONLY_ONE_EXTERNAL_DATAFIELD	There can be only one external data field for <i>value</i> .
ERR_OPEN_DIR_FAILED	Cannot open directory ' <i>value</i> '; error message ' <i>value</i> '.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_OPEN_FILE_FAILED	Cannot open file ' <i>value</i> '.
ERR_OPEN_LOG_FILE	<i>value</i> : Cannot open log file ' <i>value</i> '.
ERR_OPEN_MESSAGE_FILE	<i>value</i> : Message file ' <i>value</i> ' could not open.
ERR_OPEN_SOCIAL_FILE	Cannot open social number file ' <i>value</i> '.
ERR_OUTPUT_ALREADY_OPEN	Output stream already opened.
ERR_OUTPUT_MAPPING_FAILED	Output mapping failed: <i>value</i> .
ERR_OUTPUT_NOT_OPEN	Cannot close output stream (not open).
ERR_OUTPUT_PATH_NOT_FOUND	The output path does not exists or is not accessible.
ERR_OUTPUT_TEMP_FILE_NOT_MOVED_TO_OUTPUT	' <i>value</i> ': Cannot move temporary file ' <i>value</i> ' to output file ' <i>value</i> '.
ERR_PARAMETER_FILE_INVALID	The iRule parameter file ' <i>value</i> ' has an invalid format.
ERR_PARSE_DESCRIPTIONS	Failed to parse EDR description: <i>value</i>
ERR_PARSE_ERROR_DATA	Parse error on plug-in data: <i>value</i> .
ERR_PARSE_ERROR_STREAM	Parse error on input stream: <i>value</i> .
ERR_PCM_ERROR	PCM Error: err: <i>value</i> field: <i>value</i> loc <i>value</i> errclass: <i>value</i> rec_id: <i>value</i> resvd: <i>value</i> resvd2: <i>value</i> - <i>value</i>
ERR_PIPELINE_NOT_USABLE	The pipeline ' <i>value</i> ' is not usable; PIPELINE DEACTIVATED; check the pipeline log for error messages and start the pipeline manually.
ERR_PLUGIN_NOT_FOUND	Invalid plugin name : <i>value</i> .
ERR_PLUGIN_NOT_VALID	The module ' <i>value</i> ' is invalid and cannot be used.
ERR_PLUGIN_TYPE_INVALID	Module ' <i>value</i> ' has a wrong type.
ERR_PREFIX_DATA_NO_DELIM	Invalid delimiter count in line <i>value</i> .
ERR_PREPARE_COMMIT_TRANSACTION	Module ' <i>value</i> ' failed to prepare commit transaction ' <i>value</i> '.
ERR_PRICE_PLUGIN_INV	Price model data module invalid.
ERR_RATEPLAN_NOT_FOUND	Rateplan ' <i>value</i> ' not found in rateplan data-module.
ERR_READ_DIR_FAILED	Error reading from directory ' <i>value</i> '; error message ' <i>value</i> '.
ERR_READING_CONTRACT_PERIOD	Can't convert contract period length ' <i>value</i> ' for cli <i>value</i> .
ERR_READING_FILE	Error checking read line. Exception caught in ' <i>value</i> '; <i>value</i> ; <i>value</i> .
ERR_READING_SPECIALIST_SYSTEM	Can't convert specialist system number ' <i>value</i> ' for cli <i>value</i> .
ERR_READONLY_FILE_NOT_PROCESSED	File ' <i>value</i> ' is not writable, contents not processed.
ERR_REC_DESC_NOT_FOUND	Record description not found (<i>value</i>).

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_RECY_CANCEL_FAILED	PreRecycle: Failed to cancel transaction ' <i>value</i> '. Cannot find stream name ' <i>value</i> ' in the recycle map.
ERR_RECY_CANNOT_SET_ITEM_TYPE	PreRecycle: Cannot set the transaction item type for transaction ' <i>value</i> '.
ERR_RECY_DELETE_TMPINPUT_FILE	PreRecycle: Cannot delete temporary input file ' <i>value</i> '.
ERR_RECY_FILE_NOT_INSERT	Could not insert file name ' <i>value</i> ' into hash table.
ERR_RECY_FILE_NOT_MOVED	The file ' <i>value</i> ' could not be moved to ' <i>value</i> ' for recycling.
ERR_RECY_FILE_OPEN	The recycle database file ' <i>value</i> ' can't be opening.
ERR_RECY_FILE_WRITE	Could not write to recycle database file. Try line ' <i>value</i> ' to insert.
ERR_RECY_ROLLBACK_FAILED	PreRecycle: Failed to rollback transaction ' <i>value</i> '. Cannot move file ' <i>value</i> '.
ERR_RECYTEST_FILE_NOT_COPY	The file <i>value</i> could not copy to <i>value</i> for test recycling.
ERR_REDO_POOL_ENTRY_NOT_FOUND	Redo pool entry ' <i>value</i> ' not found in function pool.
ERR_REFERENCENAME_NOT_IN_DEF	The reference name <i>value</i> is not in the alias description.
ERR_REG_ENTRY_NOT_FOUND	Registry entry ' <i>value</i> ' not found.
ERR_REG_LOCK_FILE_EXISTS	Registry lock file ' <i>value</i> ' already exists.
ERR_REG_NAME_NOT_FOUND	Registry name ' <i>value</i> ' not found.
ERR_REG_PARSE_FAILED	Registry parse failed near ' <i>value</i> '.
ERR_REG_SUBTREE_NOT_FOUND	Registry subtree ' <i>value</i> ' not found.
ERR_REG_UPDATE_FAILED	Command processing failed for ' <i>value</i> '.
ERR_REG_VALUE_INVALID	Registry entry ' <i>value</i> ' has invalid value ' <i>value</i> '.
ERR_REG_VALUE_IS_EMPTY	Found empty value for registry item, where a value was expected.
ERR_REJECT_STREAM_NOT_DEFINED	Reject-stream not defined in ' <i>value</i> '.
ERR_RENAME_LOG_FILE	Cannot rename old logfile.
ERR_RESOLVE_STREAM_NUMBER	Failure while resolving stream number for <i>value</i> .
ERR_RETURN_PATH_NOT_FOUND	Entry for return path not found in registry.
ERR_ROLLBACK_FAILED_INPUT	Rollback failed in input-controller.
ERR_ROLLBACK_TRANSACTION	Module ' <i>value</i> ' failed to rollback transaction ' <i>value</i> '.
ERR_SCRIPT_NOT_EXE	External program (<i>value</i>) is not executable.
ERR_SCRIPT_NOT_EXIST	Cannot find external program (<i>value</i>).
ERR_SEGMENT_NOT_DEFINED	No segment defined for ' <i>value</i> '.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_SEQ_ALREADY_PROCESSED	Stream with sequence number ' <i>value</i> ' was already processed.
ERR_SEQ_CHECK_FAILED	Sequence check failed.
ERR_SEQ_ENTRY_NOT_FOUND	Cannot find entry " <i>value</i> " in the sequence control file.
ERR_SEQ_FILE_INVALID	Error reading / parsing sequence number file ' <i>value</i> '.
ERR_SEQ_GAP	Sequence number ' <i>value</i> ' is too high.
ERR_SEQ_INIT	Default sequence file generated. Check file content.
ERR_SEQ_MASTER_CONTROL	False master controller type in ' <i>value</i> ' configured.
ERR_SEQ_MASTER_CONTROLLER	Unknown or wrong master controller for sequence sharing.
ERR_SEQ_MIN_GREATER_MAX	The min sequence number is greater than the max sequence number.
ERR_SEQ_SAVE	Error saving sequence information to stream.
ERR_SETUP_CALLTYPE	Failure during setup of calltype table.
ERR_SETUP_CZT_MAPTABLE	Error while setting up CZT map table from database.
ERR_SETUP_EDRFACTORY	EDR factory setup failed: <i>value</i> .
ERR_SETUP_FSM	Failure during setup of finite state machine.
ERR_SETUP_INPUT_GRAMMAR	Input grammar setup failed: <i>value</i>
ERR_SETUP_OUTPUT	Error setup output line. Exception caught in ' <i>value</i> '; <i>value</i> ; <i>value</i> (line= <i>value</i>).
ERR_SETUP_OUTPUT_GRAMMAR	Output grammar setup failed: <i>value</i>
ERR_SHUTDOWN_FAIL_TO_COMPLETE	Shutdown request fails to finish.
ERR_SOURCE_VALUE	Source parameter must be either 'Database' or 'File'.
ERR_SPECIAL_FUNCTIONS_FAILED	The routine 'specialFunctions' in pipeline <i>value</i> failed.
ERR_STR_LEAVING_THREAD	Critical stream error. Shutting down pipeline.
ERR_STREAM_NOT_FOUND	Could not create any statistic informations for this device.
ERR_STREAM_TO_EDR_FAILED	Stream to EDR conversion failed. EDR container created, but not written to input buffer.
ERR_SYSCATALOG_ENTRY_NOT_FOUND	System catalog entry ' <i>value</i> ' not found
ERR_SYSTEM_ERROR	Unexpected error, <i>value</i>
ERR_TAM_ABORT_REQUESTED	Abort requested for transaction manager ' <i>value</i> '.
ERR_TAM_ENTRY_NOT_FOUND	Cannot find entry " <i>value</i> " in the transaction manager map.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_TAM_ENTRY_NOT_REMOVED	Cannot remove entry " <i>value</i> " from the transaction manager map.
ERR_TAM_FILE_READ_ERR	Error reading from binary transaction log file ' <i>value</i> ', message ' <i>value</i> '.
ERR_TAM_FILE_WRITE_ERR	Error writing into binary transaction log file ' <i>value</i> ', message ' <i>value</i> '.
ERR_TAM_INIT_FAILED	Failed to init transaction manager ' <i>value</i> '.
ERR_TAM_STREAM_NOT_FOUND	Cannot find stream name " <i>value</i> " for transaction id " <i>value</i> ".
ERR_TAP3_FATAL	TAP3 Fatal: Field Name: <i>value</i> , Tag: <i>value</i> , Error Code: <i>value</i> , Description: <i>value</i>
ERR_TAP3_RET	TAP3 return: Severity: <i>value</i> , Error Code: <i>value</i> , Tag: <i>value</i> , Depth: <i>value</i> , Offset: <i>value</i> , Array ID: <i>value</i> , Operator Message: <i>value</i> , Rule ID: <i>value</i> .
ERR_TAP3_SEVERE	TAP3 Severe: Field Name: <i>value</i> , Tag: <i>value</i> , Error Code: <i>value</i> , Description: <i>value</i>
ERR_TAP3_WARNING	TAP3 Warning: Field Name: <i>value</i> , Tag: <i>value</i> , Error Code: <i>value</i> , Description: <i>value</i>
ERR_TARIFF_PLUGIN_INV	Tariff model data module invalid.
ERR_TEMP_FILE_NOT_MOVED	Cannot move temporary input file ' <i>value</i> '.
ERR_THREAD_EXCEPTION	Exception detected in ' <i>value</i> '; <i>value</i> ; <i>value</i> .
ERR_THREAD_STACKSET_FAILED	Failed to set stack size of thread
ERR_TIME_PLUGIN_INV	Time model data module invalid.
ERR_TMPFILE_NOT_MOVED	Temporary file ' <i>value</i> ' could not be moved to ' <i>value</i> '.
ERR_TOKEN_ACCESS_TIMEOUT	Timeout while accessing HA role mediator token for read or update.
ERR_TOKEN_READ_FAILED	Failed to read HA role mediator token.
ERR_TOKEN_UPDATE_FAILED	Failed to update HA role mediator token.
ERR_TRACE_START_POINT_NOT_FOUND	TraceStartPoint not found.
ERR_TRACEPOINTS	TraceEndPoint is less than TraceStartPoint.
ERR_TRANS_ID_REG_ENTRY_NOT_FOUND	Registry entry ' <i>value</i> ' not found in transaction id information file ' <i>value</i> '.
ERR_TRANS_ID_REG_INVALID_VALUE	Invalid value ' <i>value</i> ' for ' <i>value</i> ' in transaction id information file ' <i>value</i> '.
ERR_TRANSFER_CUTOFF_VIOLATED	TransferCutOff Date (<i>value</i>) violated with <i>value</i> .
ERR_UNKNOWN_ALIGNMENT	Unknown alignment text in <i>value</i> . It set to left.
ERR_UNKNOWN_COL_TYPE	Unknown colType (<i>value</i>) in section info.
ERR_UNKNOWN_DEFAULT_SENTENCE	Output Stream <i>value</i> : The default line couldn't be identified. Check your format description.
ERR_UNKNOWN_DISCARD_FKT	Valid functions are [Discard or Skip]

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_UNKNOWN_EVENT_TYPE	Event <i>value</i> has unknown event type.
ERR_UNKNOWN_FIELD_NAME	Unknown field name (<i>value</i>) in section info.
ERR_UNKNOWN_ROW_TYPE	Unknown rowType (<i>value</i>) in section info.
ERR_UNKNOWN_SPLITTING_RULES	Unknown type of splitting rules <i>value</i> .
ERR_USR_PROCESS_KILLED	Killed external process ' <i>value</i> ' after it timed out.
ERR_VALUE_CONV_FAIL	Error converting value(s): <i>value</i> .
ERR_VERSION_CHECK_FAILED	Version check for database ' <i>value</i> ' and ' <i>value</i> ' failed.
ERR_WRITE_DEF_EDR_NOT_FOUND	Registry entry 'WriteDefaultEdr' not found.
ERR_WRITE_FILE	Cannot create/write file ' <i>value</i> '.
ERR_WRONG_TOKEN_COUNT	Wrong token count in input file. Line: <i>value</i>
ERR_XML_INPUT_MAPPING_FAILED	EDR XML generation failed: Input mapping ' <i>value</i> ' failed: <i>value</i> .
ERR_XML_PARSE_EDR	Exception parsing XML: near Attribute: <i>value</i> : <i>value</i>
ERR_XML_PARSE_SAX	Exception parsing XML: Line: <i>value</i> Column: <i>value</i> : <i>value</i>
ERR_XML_PARSE_UNKNOWN	Unknown exception parsing XML
ERR_XML_PARSE_XML	Exception parsing XML: <i>value</i>
ERR_ZONE_PLUGIN_INV	Zone model data module invalid.
ERR_ZONEENTRY_NOT_FOUND	Cannot find entry in zone model ' <i>value</i> ' for origin ' <i>value</i> ', destin ' <i>value</i> ', call date ' <i>value</i> ' and service ' <i>value</i> '.
ERR_ZONEMODEL_NOT_CONFIGURED	Zone model ' <i>value</i> ' has not been configured.
ERR_ZONEMODEL_NOT_FOUND	Zonemodel-Id (<i>value</i>) not found in zone data-module.
ERR_ZONETREE_NOT_FOUND	Cannot find digit tree in configuration data for zone model ' <i>value</i> '.
FORMAT_DESC_IS_INCOMPLETE	The format description is incomplete (HEADER, DETAIL, TRAILER).
INVALID_FORMAT_DESC	The format description for ' <i>value</i> ' is invalid.
UNKNOWN_LOGLEVEL	The specified log level is unknown. Valid values are normal, warning, minor, major and critical.
WRN_CANNOT_DETERMINE_OUTSTREAMNAME	Cannot determine the output file name for streamname ' <i>value</i> '.
WRN_CCENTRY_INVALID	Invalid call class map entry: <i>value</i> .
WRN_CLI_NOT_FOUND	Cli <i>value</i> not found.
WRN_CONTRACT_NOT_FOUND	Contract <i>value</i> not found.
WRN_CZTENTRY_INVALID	Invalid CZT map entry: <i>value</i> .
WRN_DEST_CLI_NOT_FOUND	Destination cli <i>value</i> not found.

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
WRN_EQUAL_TARIFFIND_DATE	Both tariff indicators have same date for contract <i>value</i> .
WRN_FILE_REMOVE_OS	<i>value</i> : Cannot remove file ' <i>value</i> '.
WRN_ILLEGAL_SPECIALDAYRATE	Illegal values in special dayrate <i>value</i> .
WRN_INVALID_ACTIVATED_DATE	Invalid activation date, ignoring contract <i>value</i> .
WRN_INVALID_CLI	Ignoring invalid cli <i>value</i> .
WRN_INVALID_CLI_RANGE	Ignoring invalid cli range (<i>value</i>).
WRN_INVALID_HISTORY_DATE	Contract ' <i>value</i> ' has an invalid history date ' <i>value</i> ', using ' <i>value</i> '
WRN_NO_ENDTRANSACTION	A beginTransaction arrives before the endTransaction in <i>value</i> .
WRN_NO_SEQUENCE_NUMBER_ADDEDTO_TRANSACTION	No new sequencenumber generated for sequence.
WRN_NO_STREAMLOG_DEFINED	Stream log not defined.
WRN_NO_VALID_ENTRY	Entry ' <i>value</i> ' in file ' <i>value</i> ' is invalid and ignored.
WRN_REG_ENTRY_OBSOLETE	Obsolete registry entry: <i>value</i>
WRN_SEMAPHORE_NOT_PROCESSED	Semaphore was not processed; check spelling.
WRN_TXNLOGGING_OFF	Transaction logging is off, make sure that you are doing testing only!
WRN_ZONEMAP_INVALID	Invalid zone map entry: <i>value</i> .
ERR_UNLINK_FILE_ERROR	Error value while attempting to unlink of temp file: value
ERR_OPEN_FILE_ERROR	Error value while attempting to open of temp file: value
ERR_WRITE_FILE_ERROR	Error value while attempting to write of temp file: value
ERR_CLOSE_FILE_ERROR	Error value while attempting to close of temp file: value
ERR_RENAME_FILE_ERROR	Error value while attempting to rename of temp file: value
ERR_TXN_TIMEOUT	Timeout while waiting for the next request in a transaction: value
ERR_RELEASE_OBJ_LOCK	Error while releasing lock for object: value
ERR_REPLENISH_POID_CACHE_FAILED	Error while processing poid: value.
ERR_PROCESS_EXIT	Attempt to exit process due to signal.
ERR_DELETION_ASS_CBD_FAILURE	Failure in deletion of ASS_CBD block.
ERR_DELETION_CP_FAILURE	Failure in deleting of CP block.
ERR_DELETION_TP_FAILURE	Failure in deleting of TP block.
ERR_CONNECT_REJECTED	Connect from ' <i>value</i> ' rejected

Table 19–1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_INCORRECT_FILE_NAME_SPECIFICATION	Error encountered in building the output file name from the given specification: 'value' for the input file - 'value'. Defaulting to regular file naming technique for this file .
ERR_IGNORE_REGISTRY_ENTRY	Registry entry - Name : 'value' Value : 'value' is ignored value
ERR_START_OVERLOAD_DETECTION	Failed to start/restart overload detection, value.
ERR_ZONE_VALUE_NOT_FOUND	ZoneValue not found for ZM-Id=value, Date=value, SC=value, A#=value, B#=value.
ERR_SESSION_PUT_ON_HOLD	Session value is put on hold due to being passive.
ERR_SESSION_REJECTED	Session value is rejected due to being passive.

Pipeline Manager Module Error Messages

Table 19–2 lists the DAT_AccountBatch error messages.

DAT_AccountBatch

Table 19–2 DAT_AccountBatch Error Messages

Error Message	Description
ERR_ACCOUNT_DB_UPDATE_FAILED	Database update failed for account (<i>value</i>) at time (<i>value</i>).
ERR_ACCOUNTBUSY_ACCOUNT_OBJ_FIELD_NOT_FOUND	Busy account obj field not found in flist for job: <i>value</i>
ERR_ACCOUNTBUSY_BATCH_OBJ_NOT_FOUND	Batch obj not found in flist for job: <i>value</i>
ERR_ACCOUNTBUSY_BATCH_OBJ_NOT_FOUND	Batch obj not found in flist for job: <i>value</i> .
ERR_ACCOUNTBUSY_JOB_ALREADY_REMOVED	Busy job with the id does not exists: <i>value</i> .
ERR_ACCOUNTBUSY_JOB_ALREADY_REMOVED	Busy job with the id does not exists : <i>value</i>
ERR_ACCOUNTBUSY_JOB_ID_FIELD_NOT_FOUND	Busy job field not found in flist: <i>value</i> .
ERR_ACCOUNTBUSY_JOB_ID_FIELD_NOT_FOUND	Busy job field not found in flist: <i>value</i>
ERR_BAD_VALUE	Null object poind in AddOrderedBalanceGroup event
ERR_BALANCE_GR_NOT_FOUND	Balance group not found for given ID
ERR_BALANCE_GROUP_UPDATE	Customer balance group update error (<i>value</i>).
ERR_BILL_INFO_UPDATE	Customer billinfo update error (<i>value</i>).
ERR BILLING_INFO_NOT_FOUND	Did not find billing information
ERR_CUSTOMER_ACCOUNT_NOT_FOUND	Customer Account not found (<i>value</i>).

Table 19–2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_CUSTOMER_EDR_PARSING	Customer EDR parsing error (<i>value</i>).
ERR_CUSTOMER_INVALID_ITEM_POID	Customer item POID not valid (<i>value</i>).
ERR_CUSTOMER_LOGIN_ACCOUNT_NOT_FOUND	Customer account not found after login (<i>value</i>).
ERR_CUSTOMER_LOGIN_INTERNAL_ERROR	Customer login internal error (<i>value</i>).
ERR_CUSTOMER_LOGIN_NOT_FOUND	Customer login not found (<i>value</i>).
ERR_CUSTOMER_LOGIN_NOT_VALID_FOR_TIME	Customer login not valid for time (<i>value</i>).
ERR_CUSTOMER_LOGIN_SERVICE_NOT_FOUND	Customer service not found (<i>value</i>).
ERR_CUSTOMER_NO_VALID_PRODUCT	Customer product not valid (<i>value</i>).
ERR_CUSTOMER_NO_VALID_PRODUCT_RATING	Customer product rating not valid (<i>value</i>).
ERR_CUSTOMER_SERVICE_NOT_FOUND	Customer Service not found (<i>value</i>).
ERR_DISCOUNT_DATA_STRING	Error building discount data string for service (<i>value</i>).
ERR_DUPLICATE_ACCOUNTBUSY_JOB_ADDED	Busy job with the id already exists : (<i>value</i>).
ERR_EVENT_ORDER_MISSING_IN_MEMORY_PROFILE	EventOrder Profile object is missing for account (<i>value</i>).
ERR_EVENT_ORDER_MISSING_SCRATCH_PAD_ITEM	EventOrderImpl::doUpdateEventOrderData() cannot find the ScratchPadItem with moniker (<i>value</i>) and pipeline/transaction (<i>value</i>).
ERR_FIRST_USAGE_ITEM_ALREADY_COMMITTED	First Usage product/discount with id (<i>value</i>). has been already committed in the pipeline.
ERR_FIRST_USAGE_OBJECT_ALREADY_INITIALIZED	First Usage product/discount with id (<i>value</i>) has been already used and initialized in probably same or different transaction, so not initializing the validity with current time.
ERR_GET_RANGE_ITEMS	DAT_LoginDbObject::getPoidRangeItems failure: Reason= <i>value</i>
ERR_INIT_ACCOUNTS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initAccounts method - <i>value</i>
ERR_INIT_BALANCE_GROUPS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initBalanceGroups method - <i>value</i>
ERR_INIT_BILL_INFOS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initBillInfo method - <i>value</i>
ERR_INIT_DELETED_ORDERED_BALANCE_GROUPS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initDeletedOrderedBalanceGroups method - <i>value</i>
ERR_INIT_GROUP_SHARING_CHARGES_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initGroupSharingCharges method - <i>value</i>

Table 19–2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_INIT_GROUP_SHARING_DISCOUNTS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initGroupSharingDiscounts method - <i>value</i>
ERR_INIT_GROUP_SHARING_PROFILES_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initGroupSharingProfiles method - <i>value</i>
ERR_INIT_LOGIN_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initLogins method - <i>value</i>
ERR_INIT_MAPPING_TABLES_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initMappingTable method - <i>value</i>
ERR_INIT_ORDERED_BALANCE_GROUPS_BILLINFO	Thread= <i>value</i> has inconsistent data in DAT::InitCustomerThread::initOrderedBalanceGroups method for account - <i>value</i>
ERR_INIT_ORDERED_BALANCE_GROUPS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initOrderedBalanceGroups method - <i>value</i>
ERR_INIT_PROFILES_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initProfiles method - <i>value</i>
ERR_INIT_PURCHASED_DISCOUNTS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initPurchasedDiscounts method - <i>value</i>
ERR_INIT_PURCHASED_PRODUCTS_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initPurchasedProducts method - <i>value</i>
ERR_INIT_SERVICE_CANCELLED	Thread= <i>value</i> has canceled in DAT::InitCustomerThread::initServices method - <i>value</i>
ERR_INIT_THREAD_DIED	Thread- <i>value</i> has died with an exception in DAT_InitCustomerThread::run() - <i>value</i>
ERR_INSERTING_CUST_CREATE_EVENT_ORDER_PROFILE	Unable to insert EventOrderProfile (<i>value</i>) for newly created account (<i>value</i>) during a CustCreate event.
ERR_INVALID_ENTRY_FOR_BUSINESS_PARAM	Invalid value for Business Parameter: <i>value</i> Value: <i>value</i>
ERR_INVALID_OUTPUT_STREAM	Invalid output stream (<i>value</i>).
ERR_INVALID_TYPE_CAST	Error on type cast <i>value</i> .
ERR_LISTENER_NOT_FOUND	Listener ' <i>value</i> ' not found.
ERR_LOOKUP_CONSTANT_ITEM	Cannot find requested item (<i>value</i>) in the ConstantItem Pool.
ERR_MAP_MERGE_THREAD_CANCELLED	DAT_MapMergeThread:: <i>value</i> has canceled because one child thread died with exception - <i>value</i>
ERR_MAP_MERGE_THREAD_DIED	DAT_MapMergeThread:: <i>value</i> has died with an exception: <i>value</i>
ERR_MAPPING_TABLE_UPDATE	Customer mapping table update error (<i>value</i>).

Table 19–2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_MULTI_THREAD_INIT	DAT_InitCustomerThread failed to create and start: Reason= <i>value</i>
ERR_MULTI_THREAD_MAP_MERGE	DAT_MapMergeThread failed to create and start: Reason= <i>value</i>
ERR_NOT_ENOUGH_CONNECTIONS	Not enough connections available to coincide with the "Threads" registry value. Define a Connections registry value greater than or equal to the Threads value.
ERR_OBG_RESOLVE_ID	Error in resolving OBG Id (<i>value</i>).
ERR_REQUIRED_REG_ENTRY_MISSING	A required registry entry is missing.
ERR_REQUIRED_REG_ENTRY_NOT_CONFIGURED	Entry not configured for DAT_Account: <i>value</i> .
ERR_RERATING_IN_PROGRESS	ReRating is currently running, EDR not processed : <i>value</i>
ERR_RW_DAT_ACCOUNT_EXCEPTION	CreateScratchPadItem() failed because item with resourceHash= <i>value</i> for Pipeline/Transaction= <i>value</i> already exists.
ERR_SCRATCH_PAD_ITEM_ALREADY_EXISTS	CreateScratchPadItem() failed because item with resourceHash= <i>value</i> for Pipeline/Transaction= <i>value</i> already exists.
ERR_SCRATCH_PAD_ITEM_IS_READ_DIRTY	WriteData() failed because item with resourceHash= <i>value</i> has been updated by another transaction.
ERR_SCRATCH_PAD_ITEM_LOCKED_BY_ANOTHER_TRANSACTION	Unable to Read or Write ScratchPadItem because it is locked by another transaction. ResourceHash= <i>value</i> Pipeline/Transaction= <i>value</i>
ERR_SCRATCH_PAD_ITEM_NOT_FOUND	ScratchPadItem not found. ResourceHash= <i>value</i> . Pipeline Transaction Hash= <i>value</i> . FunctionName= <i>value</i>
ERR_SCRATCH_PAD_ITEM_NULL	Attempting to call AdoptAndLock() will a NULL ScratchPadItem. Pipeline Transaction Hash= <i>value</i>
ERR_SCRATCH_PAD_NOT_FOUND	ScratchPad not found. Pipeline Transaction Hash= <i>value</i> . FunctionName= <i>value</i>
ERR_SERVICE_DB_UPDATE_FAILED	Account database update failed
ERR_SERVICE_NOT_CONFIGURED	Service not found (<i>value</i>).
ERR_SERVICE_OBJECT_NOT_FOUND	Service object not found for particular service Id : <i>value</i>
ERR_SERVICE_OBJECT_UPDATE_FAILED	Service object update failed for a particular service id. : <i>value</i>
ERR_SUBSCRIPTION_SERVICE_NOT_FOUND	Subscriptionservicenot found (<i>value</i>).

Table 19–2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_THREAD_CANCELLED	Thread= <i>value</i> has canceled because some child thread gets an error in DAT_InitCustomerThread::run(): Info= <i>value</i> An irrecoverable error occurred in one child thread during DAT_AccountBatch multithreaded initialization. When this occurs, other nonfailing threads safely shutdown and this error message is displayed. There is no immediate resolution.
ERR_THREAD_DIED_UNEXPECTED	An irrecoverable error occurred during DAT_AccountBatch multithreaded initialization. There is no immediate resolution; instead, open a help ticket.
ERR_UNKNOWN_DAT_ACCOUNT_EXCEPTION	Unknown Exception encountered in:(<i>value</i>)
ERR_UPDATE_BILLING_STATE_BAD_DATASTRING	BillInfo::updateBillingState() failed with invalid dataStringM (<i>value</i>)
WRN_EVENT_PROCESSING_FOR_BILLINFO_FAILED	BillInfo update failed for account ID : <i>value</i>
WRN_INVALID_ACCOUNT_IN_PROFILE	Warning, one or more account profiles point to an invalid account ID.
WRN_INVALID_SERVICE_IN_PROFILE	Warning, one or more service profiles point to an invalid service ID.
WRN_MODIFY_PROFILE_NO_SERVICE	Warning, unable to complete the Modify/CreateProfile event. Cannot locate the ERA's Service Object in the CustomerData ServiceMap. ServiceID = <i>value</i> .
WRN_MULTI_THREAD_MAP_MERGE	Login <i>value</i> found in multiple threads but update failed during map merge
WRN_REG_ENTRY_INVALID	Warning, Registry entry for DAT_Account is invalid <i>value</i>
WRN_SYSTEM_PRODUCT_MAP_IS_EMPTY	System product map is empty : <i>value</i>
WRN_SYSTEM_PRODUCT_NOT_FOUND	System product not found from system product map for particular product Id. : <i>value</i>

DAT_AccountRealtime

Table 19–3 lists the DAT_AccountRealtime error messages.

Table 19–3 DAT_AccountRealtime Error Messages

Error Message	Description
ERR_ACCRT_MESSAGE	Error message for DAT_AccountRealtime plugin module: ' <i>value</i> '.
ERR_NOT_IMPLEMENTED	Method not implemented in DAT_AccountRealtime plugin module: ' <i>value</i> '.

DAT_BalanceBatch

Table 19–4 lists the DAT_BalanceBatch error messages.

Table 19–4 DAT_BalanceBatch Error Messages

Error Message	Description
ERR_BALANCE_ATTACH_TRANS_MODULE	Cannot attach transaction module <i>value</i> .
ERR_BALANCE_DATABASE	Database operation failed in DAT_BalanceBatch: ' <i>value</i> '
ERR_BALANCE_DETACH_MODULE	Could not detach pipeline ' <i>value</i> ' (not attached).
ERR_BALANCE_GET_POID_RANGE	Get poidrange failure: Reason= <i>value</i>
ERR_BALANCE_INIT_THREAD_DIED	Thread- <i>value</i> has died with an exception in DAT_InitCustomerThread::run() - <i>value</i>
ERR_BALANCE_INVALID_BALANCEDATA	Invalid balance data during ' <i>value</i> '.
ERR_BALANCE_INVALID_EDRTRANSACTION	No transaction for this EDR on the transaction list in ' <i>value</i> '.
ERR_BALANCE_INVALID_STATE	Invalid transaction state, transId ' <i>value</i> '.
ERR_BALANCE_INVALID_TRANSACTION	Invalid transaction during ' <i>value</i> '.
ERR_BALANCE_INVALID_TRANSACTIONDATA	Invalid transaction data during ' <i>value</i> '.
ERR_BALANCE_LISTENER_NOT_FOUND	Listener ' <i>value</i> ' not found.
ERR_BALANCE_MERGE_THREAD_DIED	DAT_MapMergeThread:: <i>value</i> has died with an exception: <i>value</i>
ERR_BALANCE_MESSAGE	Error message for DAT_Balance plugin module: ' <i>value</i> '
ERR_BALANCE_MISSING_BALANCE_GROUP	Balance group missing.
ERR_BALANCE_PROCESS_EVENT_ERROR	Could not process event because there is an unknown error for event: <i>value</i> .
ERR_BALANCE_PROCESSING_EVENT_BEGIN	Could not begin event transaction for event id ' <i>value</i> '.
ERR_BALANCE_PROCESSING_EVENT_COMMIT	Could not commit event transaction for event id ' <i>value</i> '.
ERR_BALANCE_PROCESSING_EVENT_ROLLBACK	Could not rollback event transaction for event id ' <i>value</i> '.
ERR_BALANCE_THREAD_DIED_UNEXPECTED	Thread= <i>value</i> has unexpectedly died: Info= <i>value</i>
ERR_BALANCE_THREAD_INIT	Initial Thread for loading balance failed to create and start: Reason= <i>value</i>
ERR_BALANCE_THREAD_MERGE	Merge Thread for loading balance failed to create and start: Reason= <i>value</i>
ERR_BALANCE_TRANSACTION_MISMATCH	Transactions mismatch.
ERR_BALANCE_UPDATE_BALANCE	Error while updating the balance.
WRN_BALANCE_DEADLOCK_BTN_EDRTRANS	Deadlock between edr transactions on ' <i>value</i> '.

Table 19–4 (Cont.) DAT_BalanceBatch Error Messages

Error Message	Description
WRN_BALANCE_DEADLOCK_BTN_TRANS	Deadlock between currentpipelineId 'value' and another pipelineId 'value'.
WRN_BALANCE_GROUP_LOCKED	Processing on hold as BG is locked: value.
WRN_BALANCE_GROUP_NOT_FOUND	Balance group value not found.
WRN_BALANCE_INVALID_TRANSACTION	Current transaction is invalid or has errors during 'value'.
WRN_BALANCE_INVALID_TRANSACTION_ID	Invalid transaction Id : 'value'.
WRN_BALANCE_MERGE_THREAD	Login value found in multiple threads but update failed during balance merge
WRN_INVALID_CONSUMPTION_RULE	Invalid consumption rule value for resourceId value and balance group value

DAT_BalanceRealtime

Table 19–5 lists the DAT_BalanceRealtime error messages.

Table 19–5 DAT_BalanceRealtime Error Messages

Error Message	Description
ERR_BALRT_EXECUTING_OPCODE	Error executing the opcode: 'value'.
ERR_BALRT_GETTING_FLIST_FIELD	Error getting flist field: 'value'.
ERR_BALRT_MESSAGE	Error message for DAT_BalanceRealtime plugin module: 'value'.

DAT_ConnectionManager

Table 19–6 lists DAT_ConnectionManager error messages.

Table 19–6 DAT_ConnectionManager Error Messages

Error Message	Description
ERR_ALL_SERVER_CONNCTIONS_DOWN	All the server connections are down.
ERR_CREATE_LOGIN_FLIST_FAILED	Create Login flist failed .
ERR_INFRANET_GDD_INIT_FAILED	Can't initialize Infranet GDD (value)
ERR_INVALID_PROBE_VALUE	Incorrect probe value received for sending DPR.
ERR_LOGIN_FAILED	Login to CM (value) failed .
ERR_LOGIN_TO_CM	Login to CM failed for userid (value)
ERR_LOGOUT_TO_CM	Logout from CM failed for userid (value)
ERR_OPCODECALL_FAILED	Opcode call failed (value).
ERR_SERVER_CONNECT_FAILURE	Connection to server (value) failed, strerror is (value)
ERR_SERVER_CONNECTION_LOST	Server connection lost (value).
ERR_SERVER_RECONNECT_FAILURE	Server reconnection failed (value).

Table 19–6 (Cont.) DAT_ConnectionManager Error Messages

Error Message	Description
ERR_CLOSE_CONNECTION_FAILED	Failed to close connection for socket id: <i>(value)</i>

DAT_ConnectionPool

Table 19–7 lists the DAT_ConnectionPool error messages.

Table 19–7 DAT_ConnectionPool Error Messages

Error Message	Description
ERR_ALL_CM_CONNCTIONS_DOWN	All the CM connections are down.
ERR_CM_CONNECT_FAILURE	Connection to CM <i>(value)</i> failed, sterror is <i>(value)</i>
ERR_CM_CONNECTION_LOST	CM connection lost <i>(value)</i> .
ERR_CM_RECONNECT_FAILURE	CM reconnection failed <i>(value)</i> .
ERR_CONNECT_FAILED	Connect call failed from CM <i>(value)</i> .
ERR_CREATE_LOGIN_FLIST_FAILED	Create Login flist failed .
ERR_INFRANET_GDD_INIT_FAILED	Can't initialize Infranet GDD <i>(value)</i>
ERR_LOGIN_FAILED	Login to CM <i>(value)</i> failed .
ERR_LOGIN_TO_CM	Login to CM failed for userid <i>(value)</i>
ERR_LOGOUT_TO_CM	Logout from CM failed for userid <i>(value)</i>
ERR_OPCODECALL_FAILED	Opcode call failed <i>(value)</i> .
ERR_SYSTEM_ERROR	Unexpected error <i>(value)</i> .

DAT_Currency

Table 19–8 lists the DAT_Currency error message.

Table 19–8 DAT_Currency Error Messages

Error Message	Description
ERR_REGULAR_EXP	Error in Regular Expression Compilation, Desc : <i>value</i> .

DAT_Discount

Table 19–9 lists the DAT_Discount error messages.

Table 19–9 DAT_Discount Error Messages

Error Message	Description
ERR_ATTACH_DAT_DISCOUNT	Could not attach the account balance manager as ' <i>value</i> ' to DAT_DiscountPlugIn.
ERR_DAT_DSC_GENERIC	FATAL ERROR ' <i>value</i> ' line ' <i>value</i> ' msg ' <i>value</i> ' detail ' <i>value</i> '.
ERR_DB_CONNECT	Database connection is invalid. Possible solution is to restart DB & send reconnect signal. Error: <i>value</i>

Table 19–9 (Cont.) DAT_Discount Error Messages

Error Message	Description
ERR_DETERMINE_STEP	Could not determine the step of related resource id: 'value'.
ERR_DISCOUNT_DUPLICATE	Cannot insert new discount model 'value'.
ERR_DSC_EXCLUSION_REG_SETTING	Error in discount exclusion registry setting
ERR_DSCMISSING_DEF	'value'
ERR_DSCTIMEFRAME_DEF	'value'
ERR_EVENT_REGISTERED	Event 'value' could not be registered to DAT_Listener.
ERR_ISCRIPT_VALIDATION_FAILED	IScript validation failed. 'value'.
ERR_REGEX	Invalid regular expression 'value'.
ERR_RELOAD_EXTDATA_FAILURE	Re-Init of data in Discount Functional PlugIn or Balance Data PlugIn Failed.
ERR_RELOAD_FAILURE	Reloading discount pricing data failed.
ERR_THRESHOLDTO_SET_TO_MAX	Discount Step Threshold_To value: value is inappropriate, setting it to maximum: value"
WRN_WRONGVALUE_SET_TO_REGPARAM	Unexpected value set for registry parameter value.

DAT_ExchangeRate

Table 19–10 lists the DAT_ExchangeRate error message.

Table 19–10 DAT_ExchangeRate Error Message

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

DAT_InterConnect

Table 19–11 lists the DAT_InterConnect error messages.

Table 19–11 DAT_InterConnect Error Messages

Error Message	Description
ERR_GETTING_CIBER_OCC	value
ERR_GETTING_NETWORK_MODEL	Unknown network model: value.
ERR_GETTING_NETWORK_OPERATOR	Could not get network operator for value.
ERR_GETTING_PRODUCT_GROUP	Could not get product group for value.
ERR_LOADING_CIBER_OCC	Loading IFW_CIBER_OCC failed (value).
ERR_LOADING_ICPRODUCT	Loading IFW_ICPRODUCT failed (value).
ERR_LOADING_ICPRODUCT_CNF	Loading IFW_ICPRODUCT_CNF failed (value).
ERR_LOADING_NETWORK_MODEL	Loading IFW_NETWORKMODEL failed (value).
ERR_LOADING_NETWORK_OPERATOR	Loading IFW_NETWORK_OPERATOR failed (value).

Table 19–11 (Cont.) DAT_InterConnect Error Messages

Error Message	Description
ERR_LOADING_POI	Loading IFW_POI failed (<i>value</i>).
ERR_LOADING_SWITCH	Loading IFW_SWITCH failed (<i>value</i>).
ERR_LOADING_TRUNK	Loading IFW_TRUNK failed (<i>value</i>).
ERR_LOADING_TRUNK_CNF	Loading IFW_TRUNK_CNF failed (<i>value</i>).
ERR_SETUP_ICPRODUCT_CNF_ENTRY	Error while setting up IFW_ICPRODUCT_CNF table from database. Reason: <i>value</i> .

DAT_ItemAssign

Table 19–12 lists the DAT_ItemAssign error messages.

Table 19–12 DAT_ItemAssign Error Messages

Error Message	Description
ERR_FAILED_TO_GENERATE_MAP_TABLE	Failed to generate Tag and Type map table.
ERR_FAILED_TO_RESERVE_POID_IDS	Failed to reserve the Poid IDs.
ERR_FSM_CREATION_FAILED	Failed to get data from db or FSM creation failed <i>value</i>
ERR_INVALID_ITEM_POID_LIST	Item Poid List from DAT_Account is invalid.
ERR_NO_ITEM_TAG	Failed to get itemTag <i>value</i>
ERR_NO_TYPE_FOUND_FOR_TAG	No matching type found for given item tag.
ERR_SET_ITEM_POID_LIST_FAILED	Failed to set Item Poid List. <i>value</i>

DAT_Listener

Table 19–13 lists the DAT_Listener error messages.

Table 19–13 DAT_Listener Error Messages

Error Message	Description
ERR_CONVERTING_FLIST	FLIST string cannot be converted. ' <i>value</i> '.
ERR_CONVERTING_FLIST_TO_STR	Compact FLIST cannot be converted to string. ' <i>value</i> '.
ERR_OPENING_QUEUE	Error : Could not open the Queue errorCode: <i>value</i>
ERR_DEQUEUE_EVENT	Dequeue event exception (' <i>value</i> ').
ERR_DEQUEUE_NOT_ENABLED	Dequeueing for queue ' <i>value</i> ' is disabled.
ERR_ENQUEUE_EVENT	Enqueue event exception (' <i>value</i> ').
ERR_GETTING_FLIST_FIELD	Cannot get field ' <i>value</i> ' from FLIST.
ERR_OPENING_LOG_FILE	Fail to open the log file.
ERR_PURGE_EVENT_EXCEPT	Purging redundant events from queue ' <i>value</i> ' failed (exception = ' <i>value</i> ').
ERR_PURGE_EVENT_RET	Purging redundant events from queue ' <i>value</i> ' failed (retValue = ' <i>value</i> ').
ERR_QUEUE_NOT_FOUND	Queue ' <i>value</i> ' does not exist.

Table 19–13 (Cont.) DAT_Listener Error Messages

Error Message	Description
ERR_QUEUE_NOT_INSTALLED	Database queueing infrastructure has not been installed. This error occurs when the DAT_Listener registry value, QueueName does not exist in the table user_queues . To resolve this problem, configure the event queue. See "Installing and Configuring the Account Synchronization DM" in <i>BRM Installation Guide</i> .
ERR_RECEIVE_EVENT	Delivery of bus. event to DAT plugin failed (receiveEvent()).
ERR_STATVIEW_NO_ACCESS	Queue statistics view <i>value</i> cannot be accessed.
ERR_WRITING_LOG_FILE	Fail to write to the log file.
ERROR_REG_ENTRY_NOT_FOUND	Error: Registry entry not found for <i>value</i> .
WRN_NO_EVENTS	No events registered.

DAT_ModelSelector

Table 19–14 lists the DAT_ModelSelector error messages.

Table 19–14 DAT_ModelSelector Error Messages

Error Message	Description
ERR_DATABASE	FATAL ERROR ' <i>value</i> ' line ' <i>value</i> ' msg ' <i>value</i> ' detail ' <i>value</i> '.
ERR_DETAIL_DUPLICATE	Cannot insert new detail ' <i>value</i> '.
ERR_DETAIL_NULL	Cannot find detail ' <i>value</i> '.
ERR_DUPLICATE_INDEX	Cannot insert new index ' <i>value</i> '.
ERR_ELEM_GET_NEXT	Error getting element from the flist. Error msg : ' <i>value</i> '.
ERR_GENERIC	FATAL ERROR ' <i>value</i> ' line ' <i>value</i> ' msg ' <i>value</i> '.
ERR_INDEX_NOT_FOUND	Cannot find index ' <i>value</i> '.
ERR_MODEL_SELECTOR_DUPLICATE	Cannot insert new model selector ' <i>value</i> '.
ERR_MODELSELECTION_LISTENER_NOT_FOUND	Listener ' <i>value</i> ' not found
ERR_RELOAD_EXTDATA_FAILED	Reload of External Data Failed.
ERR_RELOAD_FAILURE	Reload Failed.
ERR_RULE_DUPLICATE	Cannot insert new rule ' <i>value</i> '.
ERR_RULE_LNK_DUPLICATE	Cannot insert new rule lnk ' <i>value</i> '.
ERR_RULE_NULL	Cannot find rule ' <i>value</i> '.
ERR_RULE_SET_DUPLICATE	Cannot insert new rule set ' <i>value</i> '.
ERR_SELECTOR_DETAIL	Selector Detail Error : (<i>value</i>)
ERR_SELECTOR_NOT_FOUND	Cannot find model selector ' <i>value</i> '.
ERR_VALUE_CONV_FAIL	' <i>value</i> '.

DAT_NumberPortability

Table 19–15 lists the DAT_NumberPortability error messages.

Table 19–15 DAT_NumberPortability Error Messages

Error Message	Description
ERR_CLOSE_NP_FILE	Error closing Number Portability data file <i>value</i> .
ERR_NUM_PORT_RELOAD	Error reloading data from the Number Portability data file <i>value</i> .
ERR_NUM_PORT_DELTALOAD	Error while appending additional Number Portability data from the file <i>value</i> .

DAT_PortalConfig

Table 19–16 lists the DAT_PortalConfig error messages.

Table 19–16 DAT_PortalConfig Error Messages

Error Message	Description
ERR_CBP_DATA_TYPE_MISMATCH	Data Type mismatch for param name <i>value</i> .
ERR_CBP_GROUP_DATA_NOT_FOUND	Could not find entry for group name <i>value</i> in Map
ERR_CBP_PARAM_DATA_NOT_FOUND	Could not find entry for group name <i>value</i> , param name <i>value</i> in Map
ERR_LOADING_CBP_DATA	Could not load CBP Data.
ERR_LOADING_OOD_DATA	Could not load OOD Data.

DAT_Price

Table 19–17 lists the DAT_Price error messages.

Table 19–17 DAT_Price Error Messages

Error Message	Description
ERR_APPEND_CONFIG	Config entry could not be appended to Pricemodel-Step.
ERR_INSERT_INTO_MAP	Cannot insert entry into memory map.
ERR_INSERT_STEP	Cannot insert Pricemodel-Step onto RUM.
ERR_INVALID_GL_ACCOUNT	Current GL/Account ' <i>value</i> ' does not match initial value ' <i>value</i> ' for PM= <i>value</i> , RES= <i>value</i> , RUM= <i>value</i> .
ERR_INVALID_REVENUE_GROUP	Current RevenueGroup ' <i>value</i> ' does not match initial value ' <i>value</i> ' for PM= <i>value</i> , RES= <i>value</i> , RUM= <i>value</i> .
ERR_PRICE_MODEL_CONFIG_NOT_FOUND	Cannot find PriceModel config from the PriceModel Step object.
ERR_PRICE_MODEL_STEP_NOT_FOUND	Cannot find PriceModel step from the RUM object.
ERR_PRICEMODEL_NOT_FOUND	Cannot find PriceModel ' <i>value</i> ' in table IFW_PRICEMODEL.
ERR_RESOURCE_LNK_NOT_FOUND	Cannot find the ResourceLnk object.

Table 19–17 (Cont.) DAT_Price Error Messages

Error Message	Description
ERR_RESOURCE_NOT_FOUND	Cannot find resource ' <i>value</i> ' in table IFW_RESOURCE.
ERR_RUM_NOT_FOUND	Cannot find the RUM from the ResourceLnk object.
WRN_NO_PRICEMODEL_STEPS	PM= <i>value</i> has no valid entries in IFW_PRICEMODEL_STEP configured. Skipped loading.

DAT_Rateplan

Table 19–18 lists the DAT_Rateplan error messages.

Table 19–18 DAT_Rateplan Error Messages

Error Message	Description
ERR_INVALID_RUM_IN_RUMGROUP	Found rum group(s) in IFW_RUMGROUP with no entry in IFW_RUMGROUPS_LNK (" <i>value</i> ").
ERR_INVALID_RUM_IN_SERVICE	Found rum group(s) in IFW_SERVICE with no entry in IFW_RUMGROUP (" <i>value</i> ").
ERR_INVALID_SPLITTING_TYPE	IFW_RATEPLAN.SPLITTING has invalid value ' <i>value</i> '. Possible values are '0', '1', '2', '3'.

DAT_Recycle

Table 19–19 lists the DAT_Recycle error messages.

Table 19–19 DAT_Recycle Error Messages

Error Message	Description
ERR_QUEUE_FILE_NOT_OPENED	Error opening or creating queue file <i>value</i> .
ERR_QUEUE_FILE_READ	Error reading queue file <i>value</i> .
ERR_QUEUE_FILE_WRITE	Error writing queue file <i>value</i> .

DAT_ResubmitBatch

Table 19–20 lists the DAT_ResubmitBatch error messages.

Table 19–20 DAT_ResubmitBatch Error Messages

Error Message	Description
ERR_CREATE_TEMP_FILE	Cannot create temporary file, Error <i>value</i> .
ERR_ENQUEUE_DATA	Error enqueueing data : <i>value</i> .
ERR_INVALID_OPERATION	Operation <i>value</i> , <i>value</i> .
ERR_OPENFILE_FAILED	Cannot open file <i>value</i> , Error <i>value</i> .
ERR_PROCESS_RESUBMIT_JOB	Error occurred while processing ResubmitJob : <i>value</i> .
ERR_REMOVE_OLD_ITEMS	Error occurred while removing already Processed Items

Table 19–20 (Cont.) DAT_ResubmitBatch Error Messages

Error Message	Description
WRN_PIPELINE_NOT_FOUND	Pipeline <i>value</i> not found for resubmitted batch <i>value</i> .
WRN_RENAME_FAILED	Cannot rename value to <i>value</i> , Error <i>value</i> .
WRN_RESUBMITINFO_NOTFOUND	ResubmitInfo not found for Pipeline <i>value</i> and File <i>value</i> .

DAT_ScenarioReader

Table 19–21 lists the DAT_ScenarioReader error messages.

Table 19–21 DAT_ScenarioReader Error Messages

Error Message	Description
ERR_DATATYPE_MISMATCH	Data type for <i>value</i> does not match value ' <i>value</i> '.
ERR_GROUPING_NOT_FOUND	Grouping <i>value</i> does not exist.
ERR_INVALID_DATATYPE	Data type <i>value</i> is invalid.
ERR_INVALID_FLUSHMODE	Flushmode <i>value</i> is invalid.
ERR_INVALID_FUNCTION	Function <i>value</i> is invalid.
ERR_INVALID_VALUE	Value ' <i>value</i> ' is invalid.
ERR_NO_CLASSITEMS	No classitems defined for class ' <i>value</i> '.

DAT_USC_Map

Table 19–22 lists the DAT_USC_Map error messages.

Table 19–22 DAT_USC_Map Error Messages

Error Message	Description
ERR_FSM_ENGINE_FAILED	FSM Engine Failed for zone model <i>value</i> .
WRN_INVALID_BITVEC_MATCH_FOR_SD	DAT_USC_Map_ZoneModel::Pattern Matching - bitVec match error for SD (<i>value</i>).
WRN_INVALID_PATH_NUM_FOR_SD	DAT_USC_Map_ZoneModel::Pattern Matching - invalid path number for SD (<i>value</i>).
WRN_INVALID_QTY_VAL	DAT_USC_Map_ZoneModel::Invalid quantity value for entry (<i>value</i>).
WRN_INVALID_TIME_FRAME	DAT_USC_Map_ZoneModel::Invalid timeframe for entry (<i>value</i>).
WRN_NO_USC_ENTRY_FOR_PATTERN.	DAT_USC_Map_ZoneModel::No usc-entries found during pattern matching.
WRN_NO_USC_ENTRY_FOR_PATTERN_AND_SD	DAT_USC_Map_ZoneModel::Pattern Matching - no usc-entries found for SD (<i>value</i>).
WRN_NO_USC_MAP_ENTRY_FOR_ZONE_MODEL	DAT::USC_Map::No Usc Entries found for zone model ID (<i>value</i>).
WRN_NO_VALID_USC_ENTRY	DAT_USC_Map_ZoneModel::No Valid USC entry mapping found.

DAT_Zone

Table 19–23 lists the DAT_Zone error messages.

Table 19–23 DAT_Zone Error Messages

Error Message	Description
ERR_INSERT_ZONEMODEL	Error inserting zone model into configuration.
ERR_INV_ZONECONFIG_LINE	Error invalid zone config line: <i>value</i> .
ERR_INV_ZONECONFIG_ROW	Error invalid values in INT_STANDARD_ZONE: ZONEMODEL= <i>value</i> , ORIGIN_AREACODE= <i>value</i> , DESTIN_AREACODE= <i>value</i> , SERVICECODE= <i>value</i> , VALID_FROM= <i>value</i> , VALID_TO= <i>value</i> , ZONE_WS= <i>value</i> , ZONE_RT= <i>value</i> , ALT_ZONEMODEL= <i>value</i> .
ERR_INVALID_BEAT	Error: Invalid value for Beat (must be greater than 0).
ERR_ZONEMODEL_NOT_IN_CONFIG	Cannot find zone model ' <i>value</i> ' in configuration data.
WRN_INVALID_AREACODE	AreaCode contains nondigit characters. Error= <i>value</i> .
WRN_DUPLICATE_SERVICETYPE	Duplicate entry for ServiceType= <i>value</i> corresponding to ServiceCode= <i>value</i> .

EXT_InEasyDB

Table 19–24 lists the EXT_InEasyDB error messages.

Table 19–24 EXT_InEasyDB Error Messages

Error Message	Description
ERR_ERROR_FILE_NOT_EXIST	Jobfile ' <i>value</i> ' does not exist for rollback.
ERR_READING_DATA_FROM_DATABASE	Error reading data from database -> do data from database and no eof.
WRN_FILE_NOT_MOVED	Jobfile couldn't moved to actual temp-file.

EXT_PipelineDispatcher

Table 19–25 lists the EXT_PipelineDispatcher error messages.

Table 19–25 EXT_PipelineDispatcher Error Messages

Error Message	Description
ERR_WRONG_INFILEMGR	Input file manager ' <i>value</i> ' is not of type EXT_InFileManager
ERR_RENAME_FILE_FAILED	Failed to rename file ' <i>value</i> '

FCT_Account

Table 19–26 lists the FCT_Account error messages.

Table 19–26 FCT_Account Error Messages

Error Message	Description
ERR_EMPTY_SERVICE_FIELD_MAP	No service -> edr field mapping entries for pipeline ' <i>value</i> ' in alias map.
ERR_JOB_RERATING_ACCOUNT	This Account is currently being Rerated.

FCT_AccountRouter

Table 19–27 lists the FCT_AccountRouter error messages.

Table 19–27 FCT_AccountRouter Error Messages

Error Message	Description
ERR_DATA_MODULE_IS_NOT_A_ROUTER	Data Module (<i>value</i>) is not configured as a Router.
ERR_DB_ROUTING_FAILED	Splitting failed (<i>value</i>).
ERR_JOB_AMT_MIGRATION	Job is under migration state and being directed.
ERR_REGISTRY_KEY_ERROR	Error found in registry key value pair (<i>value</i>).

FCT_AggreGate

Table 19–28 lists the FCT_AggreGate error messages.

Table 19–28 FCT_AggreGate Error Message

Error Message	Description
ERR_CTLFILE_NOT_CREATED	Control File <i>value</i> could not be created. Reason : <i>value</i>
ERR_DATABLOCK_NOT_FOUND	EDR datablock ' <i>value</i> ' not found.
ERR_EDR_ITERATOR_FAILURE	EDR indexes mismatch: <i>value</i> .
ERR_NO_DEPENDENT_CLASS_DEFINED	No dependent class defined for class ' <i>value</i> ' and classitem ' <i>value</i> '.
ERR_SCENARIO_NOT_DEFINED	Scenario ' <i>value</i> ' not defined.
WRN_NO_SCENARIOS_CONFIGURED	No scenarios configured.
WRN_SCENARIO_NOT_ACTIVE	Scenario ' <i>value</i> ' is not active.

FCT_APN_Map

Table 19–29 lists the FCT_APN_Map error messages.

Table 19–29 FCT_APN_Map Error Messages

Error Message	Description
ERR_GPRS_GSMW_AMBIGUITY	GPRS and GSMW extensions present. This is ambiguous. (<i>value</i>).
ERR_INIT_APN_MAPTABLE	Initialize of map table failed (<i>value</i>).
ERR_INIT_EDR_ITERATOR_CHARGE_PACKET	Failed to initialize charge packet iterator (<i>value</i>).
ERR_INIT_EDR_ITERATOR_ZONE_PACKET	Failed to initialize zone packet iterator (<i>value</i>).

Table 19–29 (Cont.) FCT_APN_Map Error Messages

Error Message	Description
ERR_RAZ_MAP_NOT_USABLE	Run after zoning map table not usable (<i>value</i>).
ERR_RAZ_MAP_TABLE_NOT_INITIALISED	Run after zoning map table not initialized (<i>value</i>).
ERR_RBZ_MAP_TABLE_INVALID	Run before zoning map table not usable (<i>value</i>).
ERR_RBZ_MAP_TABLE_NOT_INITIALISED	Run before zoning map table not initialized (<i>value</i>).

FCT_ApplyBalance

Table 19–30 lists the FCT_ApplyBalance error messages.

Table 19–30 FCT_ApplyBalance Error Messages

Error Message	Description
ERR_APPLYBAL_CANCEL_DEMANDED	EDR belongs to a cancel demanded transaction.
ERR_APPLYBAL_CANCEL_EDR	EDR transaction cancel demanded.
ERR_APPLYBAL_EDR_ITERATOR	Could not reset EDR iterator.
ERR_APPLYBAL_NO_ACCOUNT_BALANCE	Could not create/find balance for BG/Resource: <i>value</i> .
ERR_APPLYBAL_NO_PREFIX	No prefix specified for the notification file name
ERR_APPLYBAL_PLUGIN_INVALID_STATE	Required action does not fit to current state ' <i>value</i> '.
ERR_APPLYBAL_REALTIME	Apply Balance plugin not required for Realtime mode.
ERR_APPLYBAL_ROLLBACK_EDR	EDR transaction rollback demanded
ERR_NO_SUBBALIMPACT	No subbalance impact created for BG/Resource ' <i>value</i> ', discount not granted for this EDR. Make sure the resource is valid and there is a check for available resource in the configuration.

FCT_BatchSuspense

Table 19–31 lists the FCT_BatchSuspense error messages.

Table 19–31 FCT_BatchSuspense Error Messages

Error Message	Description
ERR_NO_DEFAULT_BATCH_SUSPENSE_REASON	No default batch suspense reason.
ERR_INVALID_RESUBMIT_INFO	Invalid resubmit information received for batch name <i>value</i> and pipeline <i>value</i> .
ERR_NO_BATCH_SUSPENSE_REASON	No batch suspense reason in the <code>/config/suspense_reason_code</code> object.

FCT_BillingRecord

Table 19–32 lists the FCT_BillingRecord error messages.

Table 19–32 FCT_BillingRecord Error Messages

Error Message	Description
ERR_BALANCE_NOT_FOUND	Error adding discount balance info: <i>value</i> .

FCT_CallAssembling

Table 19–33 lists the FCT_CallAssembling error messages.

Table 19–33 FCT_CallAssembling Error Messages

Error Message	Description
ERR_BAD_EDR_FILE_STREAM	std::fstream operation has failed for file: <i>'value'</i>
ERR_CALL_MISSING_FROM_DELETION_VECTOR	The rejected call <i>'value'</i> is missing from the CallDeletionVector.
ERR_CANNOT_CLEANUP_EDR_FILE	Cannot remove file: <i>'value'</i> during EDRFileManager::cleanupEDRFiles() operation.
ERR_CANNOT_CREATE_INDEX_FILE_STREAM	Unable to create the EDR index std::fstream for file: <i>'value'</i>
ERR_CANNOT_FIND_DEFAULT_OUTPUT_STREAM	Cannot lookup the defaultOutputStream during an UpgradeFlushLimit semaphore. Attempted location: <i>value</i>
ERR_CANNOT_OPEN_INDEX_FILE_STREAM	Unable to open the EDR index std::fstream for file: <i>'value'</i>
ERR_CHAIN_REFERENCE_MISSING	Chain reference is missing.
ERR_CREATING_CONTAINER_INDEX	Unable to create DETAIL container index from EDRFactory.
ERR_DELETION_MISSING_CALL_RECORD	Error, expected CallRecord missing from map during CallDeletionVector cleanup. <i>'value'</i>
ERR_DOM_EXCEPTION	DOMException caught in <i>value</i> : Message= <i>value</i>
ERR_EDR_ALREADY_CLOSED	The edr with the following chain reference is already closed: Chain reference= <i>'value'</i> , LongDurationIndicator= <i>'value'</i> , StartTimestamp= <i>'value'</i>
ERR_EDR_FILE_DOESNT_EXIST	EDRFile::initialize() failed because file: <i>'value'</i> does not exist.
ERR_EDR_FILE_NOT_INDEXED	The following file: <i>'value'</i> is referenced, but not available in the EDR file index.
ERR_EMPTY_MESSAGE	Reject Message <i>value</i> is missing arguments.
ERR_F_SEGMENT_ALREADY_RECEIVED	Error : An F segment has been already received with the same chain reference
ERR_FLUSH_LIMIT_IN_PROGRESS	Error: semaphore FlushLimit= <i>value</i> is already in progress, FlushLimit must finish before sending UpgradeFlushLimit.
ERR_INCLUSIVE_FLUSHZONE_LOGIC	Error InclusiveLogic, Errant Flushzone for ChainReference <i>'value'</i> .

Table 19–33 (Cont.) FCT_CallAssembling Error Messages

Error Message	Description
ERR_INCLUSIVE_LOGIC_FLUSHZONE_OUT_OF_POSITION	Error InclusiveLogic, Flushzone is out-of-position.
ERR_INCLUSIVE_LOGIC_TOO_MANY_ACTIVE_CALL_SECTIONS	Error InclusiveLogic, SingleElementCallSection has too many ACTIVE CallSections.
ERR_INDEX_FILE_RENAME_FAILED	Unable to rename tmp to index file: 'value' during IndexFile::commitFile() operation.
ERR_INVALID_CHAIN_REFERENCE	Could not find data for chain reference value.
ERR_INVALID_STATE_INDICATOR	State value is unexpected.
ERR_INVALID_TRANSID_IN_REJECT_MSG	The transaction id sent by FCT_Reject to FCT_CallAssembling is invalid: 'value'.
ERR_L_SEGMENT_ALREADY_RECEIVED	Error : An L segment has been already received with the same chain reference
ERR_LATEPARTIAL_EDR	Late EDR received and marked invalid; chain reference = value.
ERR_MISUSE_VIRTUAL_FUNCTION	Error, virtual function 'value' is not allowed.
ERR_MULTI_CALL_SECTION_MISSING_ELEMENTS	The MultiDataElementCallSection is missing DataElements.
ERR_NO_CHAIN_REFERENCE	Chain reference is missing in message value.
ERR_NOT_CA_WORKFILE	File: value is not a Call Assembling workfile.
ERR_PRODUCING_DEFAULT_EDR	Error: cannot produce default EDR for container 'value' during a flush operation.
ERR_RECYCLED_CALL_RECORD_MISSING	Error, CallRecord missing for recycled call, 'value'.
ERR_RECYCLED_EDR_NOT_FOUND_IN_MAP	Error, RecycleRequest failed for chain ref 'value'.
ERR_REJECT_CALL_RECORD_MISSING	Error, finding CallRecord during FCT_Reject AssemblyLogic lookup request. 'value'
ERR_REJECT_CALL_SECTION_MISSING	Error, finding CallSection during FCT_Reject notification request. ChainRef='value' StartTime='value'
ERR_REJECTED_EDR_NOT_IN_WORKFILE	The rejected edr is no longer in workfile. Chain reference= 'value', LongDurationIndicator= 'value', StartTimestamp= 'value'
ERR_RESTORE_ASSEMBLY_LOGIC_FAILED	Error, unable to restore AssemblyLogic in CallRecord::restore()
ERR_SAX_EXCEPTION	SAXException caught in value: Message=value
ERR_SPURIOUS_MESSAGE	Ignoring spurious message value.
ERR_UNABLE_TO_SERIALIZE_INDEX_ITEM	Unable to serialize an index item to the EDR index file named: 'value'
ERR_UNKNOWN_WORKFILE_VERSION	Cannot read workfile: value - because of unknown version#: value
ERR_UNUSED_EDRS_IN_PROCESS_RESULT	Error, un-used edrs remaining in the destructed ProcessResult object.

Table 19–33 (Cont.) FCT_CallAssembling Error Messages

Error Message	Description
ERR_UPGRADE_FLUSH_LIMIT_IN_PROGRESS	Error: semaphore UpgradeFlushLimit= <i>value</i> is already in progress, UpgradeFlushLimit must finish before sending FlushLimit.
ERR_UPGRADE_MODE_FAILURE	CallAssembly Error while Upgrading EDR with ChainRef= <i>value</i> and StartTime= <i>value</i>
ERR_VALID_DETAIL_DURING_FLUSH	During flush operation, the restored EDR with CHAIN_REFERENCE== <i>value</i> does not pass the isValidDetail() test.
ERR_XML_EXCEPTION	XMLException caught in <i>value</i> : Message= <i>value</i>
ERR_XML_IMPORT_FILE_MISSING	Unable to process ImportDataFromXml semaphore because supplied XML file ' <i>value</i> ' is missing.
ERR_XML_MEMORY_EXCEPTION	Xerces OutOfMemoryException caught in <i>value</i> : Message= <i>value</i>

FCT_CarrierIcRating

Table 19–34 lists the FCT_CarrierIcRating error messages.

Table 19–34 FCT_CarrierIcRating Error Messages

Error Message	Description
ERR_ICPRODUCT_INVALID	No valid entry in IFW_ICPRODUCT_RATE found for NM= <i>value</i> , NO= <i>value</i> , ICPRODUCT= <i>value</i> and Date= <i>value</i> .
ERR_ICPRODUCT_NETWORKMODEL_NOT_FOUND	Network model ' <i>value</i> ' not found.
ERR_ICPRODUCT_NOT_FOUND	IC_PRODUCT for GROUP/TR+DIR= <i>value</i> , DATE= <i>value</i> , SNW= <i>value</i> , DNW= <i>value</i> , A#= <i>value</i> , B#= <i>value</i> , C#= <i>value</i> , RecT= <i>value</i> , SCODE= <i>value</i> , SCLASS= <i>value</i> and UC= <i>value</i> not found (Reason ' <i>value</i> ').
ERR_INVALID_MODELTYPE	<i>value</i>
ERR_TRUNK_NOT_FOUND	No entry found for trunk under GSM extension.

FCT_CiberOcc

Table 19–35 lists the FCT_CiberOcc error messages.

Table 19–35 FCT_CiberOcc Error Messages

Error Message	Description
ERR_CIBEROCC_NETWORKMODEL_NOT_FOUND	The network model declared by registry parameter EdrNetworkModel cannot be found in the data module.
ERR_CIBEROCC_NOT_FOUND	A valid entry cannot be found for the source network (NM= <i>value</i> , SN= <i>value</i> , and DATE= <i>value</i>) in IFW_CIBER_OCC.
ERR_CREATE_OCC_EDR	An OCC EDR container cannot be created.

FCT_CliMapping

Table 19–36 lists the FCT_CliMapping error messages.

Table 19–36 FCT_CliMapping Error Messages

Error Message	Description
ERR_EDR_FACTORY	Error failed to get the edr factory (<i>value</i>).
WRN_CLIENTRY_INVALID	Setup cli Map entry failed (<i>value</i>).

FCT_CreditLimitCheck

Table 19–37 lists the FCT_CreditLimitCheck error messages.

Table 19–37 FCT_CreditLimitCheck Error Messages

Error Message	Description
WRN_NO_BG_OBJECT	No Balance Group Object found

FCT_CustomerRating

Table 19–38 lists the FCT_CustomerRating error messages.

Table 19–38 FCT_CustomerRating Error Messages

Error Message	Description
ERR_CUSTOMER_NOT_FOUND	No customer datablock found.
ERR_INIT_SLA_TABLE	Error during initialization from IFW_SLA table. See pipeline log for additional information.
ERR_RATEPLAN_NOT_DEFINED	No rateplan defined for customer account ' <i>value</i> '.
WRN_CUSTOMER_NOT_FOUND	No customer datablock found. Using Default-Rateplan ' <i>value</i> ' to continue.

FCT_DataDump

Table 19–39 lists the FCT_DataDump error messages.

Table 19–39 FCT_DataDump Error Messages

Error Message	Description
ERR_INSERT_EVENT	Failed to insert event <i>value</i>

FCT_Discard

Table 19–40 lists the FCT_Discard error messages.

Table 19–40 FCT_Discard Error Messages

Error Message	Description
WRN_FIELD_NOT_FOUND	EDR field ' <i>value</i> ' does not exist.

FCT_Discount

Table 19–41 lists the FCT_Discount error messages.

Table 19–41 FCT_Discount Error Messages

Error Message	Description
ERR_ACCOUNT_CANCEL	Could not cancel transaction ' <i>value</i> '.
ERR_ACCOUNT_COMMIT	Could not commit transaction ' <i>value</i> '.
ERR_ACCOUNT_COMMIT_RESTART	Could not commit transaction ' <i>value</i> ' on restart.
ERR_ACCOUNT_PREPARECOMMIT	Could not prepare commit transaction ' <i>value</i> '.
ERR_ACCOUNT_ROLLBACK	Could not rollback transaction ' <i>value</i> '.
ERR_BEGIN_DSC_TRANSACTION	Cannot start the transaction ' <i>value</i> '.
ERR_BEGIN_EDR	Cannot begin EDR transaction.
ERR_CANCEL_DEMANDED_EDR	EDR belongs to a cancel demanded transaction.
ERR_CANCEL_EDR	EDR transaction cancel demanded.
ERR_CANNOT_COMPILE_SCRIPT	Failed to compile the following IScript: ' <i>value</i> '.
ERR_COMMIT_EDR	Cannot commit EDR transaction.
ERR_CURRENCY_RESID_NOT_FOUND	Failed to find this currency from DAT::Currency: ' <i>value</i> '.
ERR_DISCOUNT_DETACH_MODULE	Could not detach pipeline ' <i>value</i> ' (not attached).
ERR_DSC_CONF_NOT_FOUND	Cannot find configuration for discount model ' <i>value</i> ', date ' <i>value</i> '.
ERR_EDR_ITERATOR	Could not reset EDR iterator.
ERR_EDRPACK_NOT_READY_DSC	Not all EDR fields needed for discounting are filled.
ERR_END_DSC_TRANSACTION	Cannot start the transaction ' <i>value</i> '.
ERR_EXPR_REF_CP	Expression referencing Charge Packet amount when there is no Charge Packet: ' <i>value</i> '.
ERR_GETTING_BG_ID	Failed to get balance group id for account: ' <i>value</i> '.
ERR_INVALID_BASE_AMOUNT	Invalid base amount value. Expression: ' <i>value</i> '.
ERR_INVALID_BASE_EXPR	Discount with no Charge Packet (Billing Time discount) cannot reference CP amount: ' <i>value</i> '.
ERR_INVALID_COND_AMOUNT	Invalid condition amount value. Expression: ' <i>value</i> '.
ERR_INVALID_DISCOUNT_TYPE	Invalid discount type ' <i>value</i> '.
ERR_INVALID_DRUM_AMOUNT	Invalid DRUM amount value. Expression: ' <i>value</i> '.
ERR_INVALID_GRANT_TYPE	The grant type ' <i>value</i> ' is invalid.
ERR_INVALID_THRESHOLD_AMOUNT	Invalid threshold_to amount value. Expression: ' <i>value</i> '
ERR_INVALID_THRESHOLD_TYPE	Invalid threshold type ' <i>value</i> '.
ERR_NO_ACCOUNT_BALANCE	Could not create/find balance for BG/Resource: <i>value</i> .

Table 19–41 (Cont.) FCT_Discount Error Messages

Error Message	Description
ERR_PLUGIN_INVALID_STATE	Required action does not fit to current state ' <i>value</i> '.
ERR_REJECT_EDR	EDR rejection demanded.
ERR_ROLLBACK_EDR	EDR transaction rollback demanded.

FCT_DroppedCall

Table 19–42 lists the FCT_DroppedCall error messages.

Table 19–42 FCT_DroppedCall Error Messages

Error Message	Description
ERR_DATA_TYPE_MISMATCH	Data type of ContinuationCallField = <i>value</i> does not match the data type of DroppedCallField = <i>value</i>
ERR_DROPPED_CALL_UPDATION_FAILED	Updation of Dropped Call entry in the memory map failed : <i>value</i>
ERR_FILE_ID_NOT_FOUND	File ID Not Found
ERR_FILE_FORMAT_INCORRECT	File Format is not correct
ERR_INSERT_INTO_MAP_FAILED	Insertion into the DroppedCallInfoMap failed : <i>value</i>
ERR_REMOVING_FILE	Could not remove file <i>value</i>
ERR_RESTORE_FAILED	Restore of the file failed
ERR_SERIALIZE_FAILED	Serialize to the file failed
ERR_UNABLE_TO_RETRIEVE_XML_ID	Unable to retrieve XML Id
ERR_VALUE_FROM_XML_ID	Unable to retrieve the value from XML Id
ERR_VALUE_PROFILE_ERA	Invalid value of DROPPED_CALL Profile ERA <i>value</i> = <i>value</i> . Profile ERA <i>value</i> is not set, So the default behavior is assumed.
WRN_FILE_NOT_FOUND	Main data file not found.

FCT_DuplicateCheck

Table 19–43 lists the FCT_DuplicateCheck error messages.

Table 19–43 FCT_DuplicateCheck Error Messages

Error Message	Description
ERR_BULKINSERTION_FAILED	Bulk Insertion Failed in the table <i>value</i> from file <i>value</i> . Error : <i>value</i> .
ERR_DELETION_FAILED	Deletion in the table <i>value</i> failed for transid : <i>transit_id</i> . Error: <i>value</i> .
ERR_DUP_IND_FIELD_TYPE	Wrong duplicate indicator field type (requires INTEGER).
ERR_FLUSH_TO_FILE	Could not flush data to file.
ERR_INSERT_MAP_FAILED	Insertion into the DupCheck Map failed from File : <i>value</i> .

Table 19–43 (Cont.) FCT_DuplicateCheck Error Messages

Error Message	Description
ERR_INSERTION_FAILED	Insertion in the table <i>value</i> failed . Error: <i>value</i> .
ERR_NO_INDEXSPACE_CONF	IndexSpaceName entry is not specified in the FCT_DuplicateCheck module in the registry. It is mandatory if the Database mode is configured.
ERR_NO_TABLESPACE_CONF	TableSpaceName entry is not specified in the FCT_DuplicateCheck module in the registry. It is mandatory if the Database mode is configured.
ERR_PROC_EXEC_FAILED	Procedure: <i>value</i> execute failed , <i>value</i> .
ERR_PROC_MISSING	Procedure: <i>value</i> does not exist.
ERR_REMOVE_FILE	Could not remove file <i>value</i> .
WRN_DUP_RECORD	Duplicate record found in the file.
WRN_EMPTY_FIELD	One key field is empty, field ignored.
WRN_MAIN_FILE_NOT_FOUND	Main data file not found.
WRN_NO_ROOT_FIELD	Defined field is not in root block.

FCT_EnhancedSplitting

Table 19–44 lists the FCT_EnhancedSplitting error messages.

Table 19–44 FCT_EnhancedSplitting Error Messages

Error Message	Description
ERR_ADD_SPLITTING_PATTERNS	Failed to add patterns to fsm (<i>value</i>): <i>value</i> .
ERR_INSERT_SPECSYS	Failed to insert ' <i>value=</i> <i>value</i> ' into the mapping table.
ERR_NO_SPLITTING_ENTRY	No matching splitting rule found.
ERR_NO_STREAM_FOR_SPECSYS	No output stream for specialist system ' <i>value</i> '.
ERR_SETUP_SPLITTING_ENTRY	Setup for splitting entry (<i>value</i>) failed: <i>value</i> .

FCT_ExchangeRate

Table 19–45 lists the FCT_ExchangeRate error messages.

Table 19–45 FCT_ExchangeRate Error Messages

Error Message	Description
ERR_EXCHANGERATE_BRK_HEADERDATE	File creation date is invalid : ' <i>value</i> '.
ERR_EXCHANGERATE_FILEDATE_NOT_EXIST	File date does not exist: ' <i>value</i> '.
ERR_EXCHANGERATE_NOT_FOUND	Exchangerate not found: ' <i>value</i> ', ' <i>value</i> ' From_Currency: ' <i>value</i> ', To_Currency: ' <i>value</i> ' .

FCT_Filter_Set

Table 19–46 lists the FCT_Filter_Set error messages.

Table 19–46 FCT_Filter_Set Error Messages

Error Message	Description
ERR_INVALID_DISCOUNT_ID	Invalid discount object id ' <i>value</i> '.
ERR_INVALID_PRODUCT_ID	Invalid product id ' <i>value</i> '.

FCT_IRules

[Table 19–47](#) lists the FCT_IRules error messages.

Table 19–47 FCT_IRules Error Messages

Error Message	Description
ERR_ILLEGAL_CUSTOMER_KEY	Illegal customer key ' <i>value</i> '.
ERR_ILLEGAL_LICENSE_KEY	Illegal license key ' <i>value</i> '.
ERR_RULE_DESCRIPTION	Failed to create ruleset from description: <i>value</i> .
ERR_RULE_SETUP	Failed to setup rule ' <i>value</i> ': <i>value</i> .
ERR_RULESET_FILE	<i>value</i> :line <i>value</i> : <i>value</i> .
ERR_RULESET_NOT_FOUND	Ruleset ' <i>value</i> ' not found.

FCT_IScriptPlugin

[Table 19–48](#) lists the FCT_IScriptPlugin error messages.

Table 19–48 FCT_IScriptPlugin Error Messages

Error Message	Description
ERR_SCRIPT_NOT_USABLE	The iScript ' <i>value</i> ' is not usable.

FCT_ItemAssign

[Table 19–49](#) lists the FCT_ItemAssign error messages.

Table 19–49 FCT_ItemAssign Error Messages

Error Message	Description
ERR_INVALID_ITEM_POID	Invalid item poid returned from DAT_ItemAssign.

FCT_MainRating

[Table 19–50](#) lists the FCT_MainRating error messages.

Table 19–50 FCT_MainRating Error Messages

Error Message	Description
ERR_INVALID_ADDON_TYPE	Addon-Type in IFW_RATEPLAN_CNF for RP= <i>value</i> , RP-V= <i>value</i> , IC= <i>value</i> , SCode= <i>value</i> , SClass= <i>value</i> , TM= <i>value</i> and TZ= <i>value</i> is invalid. Not in 'Percentage', 'Absolute' or 'New' value.

Table 19–50 (Cont.) FCT_MainRating Error Messages

Error Message	Description
ERR_PRICE_MODEL_CONFIG_NOT_FOUND	Pricemodel-Config not found for PM= <i>value</i> , RES= <i>value</i> , RUM= <i>value</i> , Step= <i>value</i> , Frame= <i>value</i> and Date= <i>value</i> .
ERR_PRICE_MODEL_NOT_FOUND	Pricemodel ' <i>value</i> ' not found. See process-log-file for invalid pricemodels not loaded. Packet no. ' <i>value</i> '.
ERR_PRICE_MODEL_RUM_NOT_FOUND	Pricemodel-RUM not found for PM= <i>value</i> , RES= <i>value</i> and RUM= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_PRICE_MODEL_STEP_NOT_FOUND	Pricemodel-Step not found for PM= <i>value</i> , RES= <i>value</i> RUM= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_RATE_PRICE_MODEL_NOT_FOUND	Pricemodel not found (IFW_RATEPLAN_CNF) for RP= <i>value</i> , RPV= <i>value</i> , IC= <i>value</i> , SCode= <i>value</i> , SClass= <i>value</i> , TMM= <i>value</i> and TZ= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_RATEPLAN_VERSION_DATE_NOT_FOUND	Rateplan-Version not found (IFW_RATEPLAN_VER) for Rateplan= <i>value</i> and Date= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_RATEPLAN_VERSION_ID_NOT_FOUND	Rateplan-Version not found (IFW_RATEPLAN_VER) for Rateplan= <i>value</i> and Version= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_RUM_GROUP_NOT_FOUND	Found no valid rum group for packet no. ' <i>value</i> '.
ERR_TIME_MODEL_NOT_FOUND	Timemodel not found (IFW_RATEPLAN_CNF) for RP= <i>value</i> , RPV= <i>value</i> , IC= <i>value</i> , SCode= <i>value</i> and SClass= <i>value</i> . Packet no. ' <i>value</i> '.
ERR_TIMEZONE_NOT_FOUND	Timezone not found in TimeModel= <i>value</i> for Date= <i>value</i> . Packet no. ' <i>value</i> '.

FCT_Opcode

Table 19–51 lists the FCT_Opcode error messages.

Table 19–51 FCT_Opcode Error Messages

Error Message	Description
ERR_ALL_CM_CONNECTIONS_LOST	All CM connections lost.
ERR_RECV_DATA	Error while receiving data from CM (<i>value</i>)
ERR_RESTORE_DATA	Error while restoring data (<i>value</i>)
ERR_SEND_DATA	Error occurred while sending data to CM (<i>value</i>)
ERR_SERIALIZE_EDR	Exception occurred while serializing edr (<i>value</i>)

FCT_PreRating

Table 19–52 lists the FCT_PreRating error messages.

Table 19–52 FCT_PreRating Error Messages

Error Message	Description
ERR_ASS_CBD_NOT_FOUND	No ASSOCIATED_CHARGE_BREAKDOWN block is found
ERR_NO_RATEPLAN	No rateplan-code or -id defined in DETAL.ASS_CBD.CP.
ERR_RATEPLAN_NOT_A_NUMBER	Rateplan-Id ' <i>value</i> ' is not a number.
ERR_RATEPLAN_TYPE_INV	Rateplan-Type ' <i>value</i> ' does not match valid values ('R').
ERR_RATEPLAN_VERSION_NOT_FOUND	Rateplan-Version not found for Id (<i>value</i>) and Date (<i>value</i>).
ERR_ZONE_VALUE_NOT_FOUND	ZoneValue not found for ZM-Id= <i>value</i> , Date= <i>value</i> , SC= <i>value</i> , A#= <i>value</i> , B#= <i>value</i> .

FCT_PreSuspend

Table 19–53 lists the FCT_PreSuspend error messages.

Table 19–53 FCT_PreSuspend Error Messages

Error Message	Description
ERR_DATATYPE_NOT_SUPPORTED	Data type EDR::FieldDescr:: <i>value</i> is not supported.
ERR_NO_QUERYABLE_FIELDS_TABLE	No queryable fields specified in registry.
WRN_NO_QUERYABLE_FIELDS	No queryable fields specified for table <i>value</i> .

FCT_RateAdjust

Table 19–54 lists the FCT_RateAdjust error message.

Table 19–54 FCT_RateAdjust Error Message

Error Message	Description
ERR_ADD_RATEADJUST	Failure while adding rate adjust entry (<i>value</i>).

FCT_Reject

Table 19–55 lists the FCT_Reject error message.

Table 19–55 FCT_Reject Error Message

Error Message	Description
ERR_REJECT_UNKNOWN_STREAM	Error : Stream ' <i>value</i> ' is unknown.

FCT_Rounding

Table 19–56 lists the FCT_Rounding error message.

Table 19–56 FCT_Rounding Error Message

Error Message	Description
ERR_APP_NAME_NOT_FOUND	The Application Type name <i>value</i> provided for the Registry Entry <i>value</i> is not supported.

FCT_SegZoneNoCust

[Table 19–57](#) list the FCT_SegZoneNoCust error message.

Table 19–57 FCT_SegZoneNoCust Error Message

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

FCT_Suspense

[Table 19–58](#) lists the FCT_Suspense error messages.

Table 19–58 FCT Suspense Error Messages

Error Message	Description
ERR_ASS_SUSPENSE_EXT_MISSING	No ASS_SUSPENSE_EXT data block. FCT_PreSuspense is not active.
ERR_EFENTRY_INVALID	Failed to setup suspense mapping entry (<i>value</i>)
ERR_NO_DEFAULT_SUSPENSE_REASON	No default suspense reason and subreason specified.
ERR_NO_MATCHING_ENTRY_IN_EDR_FLD_MAP	No matching entry found in suspense edr field map table.
ERR_REJECT_STREAM_ERROR	RejectStream registry must be set to “ <i>value</i> ”.
WRN_FIELD_MAP_NOT_FOUND	Registry entry ‘ <i>value</i> ’ not found.

FCT_Timer

[Table 19–59](#) lists the FCT_Timer error messages.

Table 19–59 FCT_Timer Error Messages

Error Message	Description
ERR_CAN_NOT_SCHEDULE_TIMER	Can't schedule timer.
ERR_CAN_NOT_RESCHEDULE_TIMER	Can't reschedule timer id ' <i>value</i> '.
ERR_CAN_NOT_RESET_TIMER	Can't reset timer id ' <i>value</i> '.
ERR_CAN_NOT_CANCEL_TIMER	Can't cancel timer id ' <i>value</i> '.

FCT_TriggerBill

[Table 19–60](#) lists the FCT_TriggerBill error message.

Table 19–60 FCT_TriggerBill Error Message

Error Message	Description
ERR_TRIGGER BILLING	TriggerBilling is required.

FCT_UoM_Map

[Table 19–61](#) lists the FCT_UoM_Map error messages.

Table 19–61 FCT_UoM_Map Error Messages

Error Message	Description
ERR_INIT_PBC_SERVICE_MAP_TABLE	Failed to init. service map table (<i>value</i>).
ERR_INIT_PBC_SERVICE_RGL_MAP_TABLE	Failed to init. rgl map table (<i>value</i>).
ERR_INIT_UoM_MAPTABLE	Failed to initialise the UoM map table (<i>value</i>).
ERR_NO_SERVICE_CODE_NAME_SUPPLIED	No service code value was supplied for the mapping (<i>value</i>).

FCT_UsageClassMap

[Table 19–62](#) lists the FCT_UsageClassMap error message.

Table 19–62 FCT_UsageClassMap Error Message

Error Message	Description
ERR_UCENTRY_INVALID	Failed to setup usage class mapping entry (<i>value</i>).

FCT_USC_Map

[Table 19–63](#) lists the FCT_USC_Map error messages.

Table 19–63 FCT_USC_Map Error Messages

Error Message	Description
ERR_SETUP_USC_MAPTABLE	Failed to initialize the USC map table (<i>value</i>).
ERR_USC_GROUP_VALUE_NOT_FOUND	No specified for USC group registry parameter.

INP_GenericStream

[Table 19–64](#) lists the INP_GenericStream error message.

Table 19–64 INP_GenericStream Error Message

Error Message	Description
ERR_INP_GENERICSTREAM_PARSE_ERROR_STREAM	Parse error on input stream: <i>value</i> .

INP_Realttime

[Table 19–65](#) lists the INP_Realttime error messages.

Table 19–65 INP_Realttime Error Messages

Error Message	Description
ERR_REALTIME_BLOCKINDEX_NOT_IN_FACTORY	RealtimePipeline: Container BlockIndex is missing in Factory: <i>value</i>
ERR_REALTIME_CANNOT_CREATE_DOMBUILDER	RealtimePipeline: Unable to create a DOMBuilder parser from DOMImplementationLS object.
ERR_REALTIME_CANNOT_DUPLICATE_EDR	RealtimePipeline: Unexcepted to duplicate EDR.

Table 19–65 (Cont.) INP_Realtime Error Messages

Error Message	Description
ERR_REALTIME_COMPILE_ISCRIPT_FAILED	RealtimePipeline: Unable to compile iscript mapping file: <i>value</i>
ERR_REALTIME_CREATE_DEFAULT_EDR_FAILED	RealtimePipeline: Unable to create a default edr for CM error propagation.
ERR_REALTIME_CREATING_CONTAINER_INDEX	RealtimePipeline: Unable to create EDR Container Index for container name: <i>value</i>
ERR_REALTIME_EDR_FIELD_MAPPING_MISMATCH	RealtimePipeline: EDRField type mismatch error, unable to map value: <i>value</i> to EDR field type: <i>value</i> in function readConstants()
ERR_REALTIME_EDR_TYPE_MISMATCH	RealtimePipeline: Type mismatch error, unable to map pin_fld_t: <i>value</i> to EDR field: <i>value</i> in function appendValueToEDR()
ERR_REALTIME_EXECUTE_ISCRIPT_FAILED	RealtimePipeline: Failed to execute realtime iscript.
ERR_REALTIME_INDEX_NOT_IN_FACTORY	RealtimePipeline: Container Index is missing in Factory: <i>value</i>
ERR_REALTIME_INSERT_FLIST_EXT	RealtimePipeline: IScript Flist extension is missing.
ERR_REALTIME_MISSING_INPUT_MAP_LIST	RealtimePipeline: XML element list <InputMap> missing in xml file: <i>value</i>
ERR_REALTIME_MISSING_INPUT_MAP_NODE	RealtimePipeline: XML element <InputMap> missing in xml file: <i>value</i>
ERR_REALTIME_MISSING_OPCODE_LIST	RealtimePipeline: XML element list <OpcodeMap> missing in xml file: <i>value</i>
ERR_REALTIME_MISSING_OPCODE_NODE	RealtimePipeline: XML element <OpcodeMap> missing in xml file: <i>value</i>
ERR_REALTIME_MISSING_REQUIRED_FLIST_FIELD	RealtimePipeline: The expected Input Flist field is missing: <i>value</i>
ERR_REALTIME_NULL_EDR_FACTORY	RealtimePipeline: Null EDR factory returned from edrFactory()
ERR_REALTIME_OBS_FLIST_EDR_CONVERSION_FAILED	Realtime Pipeline: Conversion of Input flist to EDR failed.
ERR_REALTIME_OPEN_ISCRIPT_FAILED	RealtimePipeline: Unable to open iscript mapping file: <i>value</i>
ERR_REALTIME_PRODUCING_EDR	RealtimePipeline: Unable to produce EDR Container from index named: <i>value</i>
ERR_REALTIME_PROPAGATE_BAD_DATABLOCK_VALUE	RealtimePipeline: PropagateBlock blockname: <i>value</i> , unable to find target EDR::DatablockValue at index [<i>value,value</i>]
ERR_REALTIME_PROPAGATE_BLOCK_CLONE	RealtimePipeline: PropagateBlock blockname: <i>value</i> , unable to clone source EDR::Datablock at index <i>value</i>
ERR_REALTIME_PROPAGATE_BLOCK_INVALID_ARG	RealtimePipeline: PropagateBlock blockname: <i>value</i> is not a valid candidate for propagation.
ERR_REALTIME_PROPAGATE_BLOCK_INVALID_DEPTH	RealtimePipeline: PropagateBlock blockname: <i>value</i> has an invalid depth for propagation.

Table 19–65 (Cont.) INP_Realtime Error Messages

Error Message	Description
ERR_REALTIME_PROPAGATE_BLOCK_LOOKUP_SOURCE_BLOCK	RealtimePipeline: PropagateBlock blockname: <i>value</i> has missing source EDR::Datablock at index <i>value</i>
ERR_REALTIME_PROPAGATE_BLOCK_MISSING	RealtimePipeline: PropagateBlock blockname: <i>value</i> is not a member of the current EDR.
ERR_REALTIME_PUSH_EDR_FAILED	RealtimePipeline: Failed to push EDRs onto InputDevice.
ERR_REALTIME_READ_REGISTRY_FAILED	RealtimePipeline: ReadRegistry() failed for INP::Realtime module.
ERR_REALTIME_REG_INTERPRETER	RealtimePipeline: IScript interpreters missing.
ERR_REALTIME_SET_INTERNAL_TRANSACION_ID_FAILED	RealtimePipeline: Failed to set Transaction Id in Internal block.
ERR_REALTIME_UNKNOWN_EXCEPTION	RealtimePipeline: Unknown exception in function: <i>value</i>

INP_Recycle

Table 19–66 lists the INP_Recycle error messages.

Table 19–66 INP_Recycle Error Messages

Error Message	Description
ERR_ADD_HEADER_REOCD_ERROR	Error while adding header record.
ERR_ADD_TRAILER_REOCD_ERROR	Error while adding trailer record.
ERR_INP_RECYCLE_CANNOT_CONVERT_EDR	Cannot convert EDR to new container desc, field is either missing or type has changed: <i>value</i> .
ERR_INP_RECYCLE_CANNOT_FIND_EDR_VERSION	INP::EdrVersionConverter::getEdrVersion(..) cannot find the EDR version attribute in the input xml.
ERR_INP_RECYCLE_DOM_PARSE_ERRORS	DOMParser errors encountered during parse operation in function <i>value</i> .
ERR_INP_RECYCLE_EDR_MISSING_FM	EDR xml missing FM_ELEMENT in function <i>value</i> .
ERR_INP_RECYCLE_EMPTY_QUERY_RESULT	Query for edr_fld_map_buf_t buffer returned empty data in function: <i>value</i> .
ERR_INP_RECYCLE_INVALID_READER	Invalid reader in function: <i>value</i> - <i>value</i> .
ERR_INP_RECYCLE_NO_DB_CONNECTION	Unable to get DB Connection in function: <i>value</i> - <i>value</i> .
ERR_INP_RECYCLE_NO_DB_SELECTOR	Unable to get DB Selector in function: <i>value</i> - <i>value</i> .
ERR_INP_RECYCLE_NO_DB_TABLES	Unable to get DB Tables edr_field_mapping_t and edr_fld_map_buf_t in function: <i>value</i> .
ERR_INP_RECYCLE_ROOT_ELEMENT_MISSING	EDR root XML element not found in function <i>value</i> .
ERR_INP_RECYCLE_SAX2_PARSE_ERRORS	SAX2XMLReader errors encountered during parse operation in function <i>value</i> .

Table 19–66 (Cont.) INP_Recycle Error Messages

Error Message	Description
ERR_INP_RECYCLE_UNEXPECTED_EXCEPTION	Unexpected Exception in function <i>value</i> .
ERR_PARSE_HEADER_ERROR	Error in parsing header record.
ERR_PROCESS_RECORD_ERROR	Error in processing record : <i>value</i> .
ERR_PUSH_EDR_ERROR	Error in pushing record to pipeline,record number: <i>value</i> .
ERR_RECYCLING_DATA_MODULE_NOT_FOUND	Cannot find recycling data module.
ERR_REG_TRAILER_NOT_SUPPORTED	Trailer record is not supported by INP_Recycle.

NET_EM

Table 19–67 lists the NET_EM error messages.

Table 19–67 NET_EM Error Messages

Error Message	Description
ERR_CLOSE_REALTIME_TRANSACTION	Failed to close realtime transaction in pipeline ' <i>value</i> '.
ERR_CREATE_CONTEXT_FAILED	PCM_CREATE_CONTEXT failed for socket ' <i>value</i> '.
ERR_LSOCK_BIND	UNIX Sock bind error for file name ' <i>value</i> '.
ERR_OPCODE_NOT_CONFIGURED	Opcode ' <i>value</i> ' is not configured.
ERR_OPEN_REALTIME_TRANSACTION	Failed to open realtime transaction in pipeline ' <i>value</i> '.
ERR_RTP_ARE_NOT_READY	All Realtime Pipelines NOT ready.
ERR_SOCKET_ACCEPT	Accept failed for socket ' <i>value</i> ', errno ' <i>value</i> '.
ERR_SOCKET_BIND	TCP/IP Socket bind error ' <i>value</i> ', errno ' <i>value</i> '.
ERR_SOCKET_LISTEN	Listen failed for socket ' <i>value</i> ', errno ' <i>value</i> '.
ERR_SOCKET_BIND	Socket bind error .

OUT_DB

Table 19–68 lists the OUT_DB error messages.

Table 19–68 OUT_DB Error Messages

Error Message	Description
ERR_DB_ROLLBACK_TRANSACTION	Database transaction collback failed for stream ' <i>value</i> '.
ERR_DB_STREAM_CLOSE	Database could not closed for stream ' <i>value</i> '.
ERR_DB_STREAM_OPEN	Database open failed for stream ' <i>value</i> '.
ERR_FILE_NOT_CONSISTENT	Parameter ' <i>value</i> ' file not consistent.
ERR_INDEXLIST_NOT_CREATED	Could not create index table for edr-container fields.

OUT_GenericStream

Table 19–69 lists the OUT_GenericStream error messages.

Table 19–69 OUT_GenericStream Error Messages

Error Message	Description
ERR_OUTPUT_PARSE_ERROR	Parse error on output file: <i>value</i> .
ERR_STREAM_IS_EMPTY_RETURN	Function streamIsEmpty() needs boolean return type.

OUT_Realtime

Table 19–70 lists the OUT_Realtime error messages.

Table 19–70 OUT_Realtime Error Messages

Error Message	Description
ERR_OUT_REALTIME_CREDIT_LIMIT_CHECK_FAILED	RealtimePipeline: CreditLimitCheck failed.
WRN_OUT_REALTIME_REVERSE_RATING_APPLIED	RealtimePipeline: Reverse Rating Applied.

Pipeline Utility Error Messages

LoadIFWConfig

Table 19–71 lists the LoadIFWConfig error messages.

Table 19–71 LoadIFWConfig Error Messages

Error Message	Description
ERR_CREATE_OBJECT_FAILED	Cannot create object <i>value</i> (invalid (NULL) pointer).
ERR_INVALID_PATTERN	Directory pattern <i>value</i> is invalid.
ERR_NO_DIR	Directory <i>value</i> not accessible.
ERR_NO_PATH_NAME	No path name given.
ERR_OBJ_NOT_INITIALIZED	The object <i>value</i> is not initialized.
ERR_REG_NAME_NOT_FOUND	Registry name <i>value</i> not found.
ERR_REG_PARSE_FAILED	Registry parse failed near <i>value</i> .
ERR_REG_SUBTREE_NOT_FOUND	Registry subtree <i>value</i> not found.
ERR_REG_VALUE_IS_EMPTY	Found empty value for registry item, where a value was expected.
ERR_SYSTEM_ERROR	Unexpected error, errno <i>value value value</i>
WRN_REG_ENTRY_OBSOLETE	Obsolete registry entry: <i>value</i>

OMF Error Messages

Table 19–72 lists the OMF error messages.

Table 19–72 OMF Error Messages

Error Message	Description
ERR_SNMP_LOST_CONNECTION	Lost connection to master SNMP agent.
ERR_SNMP_NOT_REGISTERED	Failed to register the mib table OID <i>value</i> for probe <i>value</i> at master SNMP agent.
ERR_SNMP_NOT_UNREGISTERED	Failed to un-register the mib table OID <i>value</i> for probe <i>value</i> at master SNMP agent.
ERR_SNMP_SUBAGENT_MIB_INIT	Failed to initialize SNMP subagent mib. Registration will be reattempted periodically.

Part V

Managing IMDB Cache-Enabled Systems

Part V describes how to use Oracle In-Memory Database (IMDB) Cache Manager in your Oracle Communications Billing and Revenue Management (BRM) system. It includes the following chapters:

- [Using Oracle IMDB Cache Manager](#)
- [Installing IMDB Cache Manager](#)
- [Configuring IMDB Cache Manager](#)
- [Generating the BRM Cache Group Schema](#)
- [Customizing IMDB Cache Manager](#)
- [Migrating Data to an Oracle IMDB Cache-Enabled BRM System](#)

Using Oracle IMDB Cache Manager

This chapter provides information about using Oracle Communications Billing and Revenue Management (BRM) In-Memory Database (IMDB) Cache Manager.

About Oracle IMDB Cache Manager

IMDB Cache Manager is an optional BRM component that enables you to use both:

- **Oracle IMDB Cache:** An in-memory database for caching only performance-critical data from the BRM database.
- **BRM database:** An external relational database management system (RDBMS) for storing all BRM data.

The IMDB Cache Manager package includes only the IMDB Cache Data Manager (DM). The DM provides the interface between the CM and Oracle IMDB Cache and between the CM and the BRM database. The DM determines whether to route CM requests to Oracle IMDB Cache or to the BRM database based on where the data resides (called the *residency type*). For more information, see ["About the Residency Type and Oracle IMDB Data Manager"](#).

About Caching BRM Objects in Memory

You can cache a subset of your BRM database tables in an in-memory database by using Oracle IMDB Cache. Oracle IMDB Cache includes the following functionality:

- Caches data from BRM database tables
- Replicates cached data for high-availability systems
- Stores transient data

For more information about Oracle IMDB Cache, see "About Oracle IMDB Cache" in *BRM Concepts*.

About BRM Cache Groups

Oracle IMDB Cache stores BRM database tables in cache groups. Each cache group can store a single BRM table or a group of related tables (root table and one or more child tables). Oracle IMDB Cache synchronizes the data between the cache groups and the BRM database.

BRM cache groups store BRM data that require low latency and high throughput for fast access. For example:

- Subscriber data to process authentication, authorization, and accounting (AAA) requests
- Subscriber data accessed by customer service representatives (CSRs)
- Account, bill unit, and item data needed to perform billing

By default, the IMDB Cache Manager installer creates the following cache groups in your Oracle IMDB Cache data stores:

- A subscriber cache group. This cache group is defined as a Global ASYNCHRONOUS WRITETHROUGH (AWT) type, which makes the data available to all members in a cache grid. If the requested data is not in the subscriber cache group, Oracle IMDB Cache returns an error.
- Event cache groups. These cache groups are defined as Dynamic AWT types, in which updates in the cache group are asynchronously propagated to the BRM database. By default, data in a Dynamic AWT cache group is deleted using the least-recently-used (LRU) aging policy. If the requested data is not in an event cache group, Oracle IMDB Cache forwards the request to the BRM database, gets the response, caches the data, and sends the response to the client.

You can view the default BRM cache group definitions in the *BRM_home/bin/pin_tt_schema_gen.values* configuration file.

See *Oracle In-Memory Database Cache User's Guide* for more information about cache group types and aging policy.

Guidelines for Setting Cache Group Size

When you create an Oracle IMDB Cache data store, you specify the maximum size of the data store's shared-memory segment. Data is created in the shared-memory segment until it reaches the specified maximum. After it reaches the maximum, Oracle IMDB Cache reports an out-of-memory error any time you attempt to load data into or create accounts in the shared-memory segment.

By default, the maximum size of the data store's shared-memory segment is 32 MBytes. You can change the maximum size value at the following times by using the **PermSize** cache attribute:

- When you create the data store.
- When you connect to the data store the first time. In this case, you can only increase the maximum size, not decrease it.

For more information about the **PermSize** cache attribute, see the discussion on specifying the size of a data store in *Oracle TimesTen In-Memory Database Operations Guide*.

About Loading BRM Objects into Oracle IMDB Cache

You load BRM objects into Oracle IMDB Cache by using the following:

- The **pin_tt_schema_gen** utility, which uses the existing BRM database schema to generate your schema and load SQL scripts.
- The schema SQL script, which creates the BRM cache group schema in Oracle IMDB Cache.
- The load SQL script, which loads BRM data into the cache group schema.

For more information, see ["Generating the BRM Cache Group Schema"](#).

About Loading Migrated Accounts into the Cache Groups

If you migrated accounts into the BRM database after your BRM system is up and running, you can load the migrated data into the BRM cache groups using the **pin_tt_schema_gen** utility. To perform the migration without downtime or without blocking operations on the objects already in the cache, load only the objects that are not already cached in the Oracle IMDB Cache data store.

To reduce the impact on your system, Oracle recommends that you perform the load in batches.

About Managing Data in Oracle IMDB Cache

Database tables cached in Oracle IMDB Cache are stored in shared memory, which is a fixed size. You must manage the data in the cache to avoid running out of shared-memory space.

Managing Fast-Growing Tables

To manage fast-growing tables, such as */event* tables, in BRM cache groups, use the Oracle IMDB Cache Aging feature.

IMDB Cache DM uses *usage-based* aging to remove least-recently-used (LRU) data, based on a range of threshold values that specifies when LRU aging is activated and deactivated.

The aging policy for BRM cache groups is defined in the cache group schema definitions in the **pin_tt_schema_gen.values** file. Aging is enabled by default for Dynamic AWT cache groups. See "[Generating the BRM Cache Group Schema](#)".

About Purging Expired Reservation and Active-Session Objects

When account balances in the BRM database are updated at the end of a session, reservation objects and active-session objects are removed from Oracle IMDB Cache. The reservations are released and session objects in the database are rated. However, when a session is terminated abnormally (for example, because of a timeout error or not receiving a stop-accounting request), the session remains open and the reservation objects and the related objects in that session remain in the database cache, using up shared-memory space. You remove expired reservation and active session objects and release shared-memory space by using the **pin_clean_rsvns** and **pin_clean_asos** utilities.

For information on how to run these utilities, see "[pin_clean_rsvns](#)" and "[pin_clean_asos](#)".

About Purging Closed Bills, Items, Journals, and Expired Subbalances

You remove closed bills, items, journals, and expired account subbalances by using the **pin_purge** utility.

Note: Do not use the **pin_sub_bal_cleanup** utility to purge expired account subbalances. Using the **pin_sub_bal_cleanup** utility results in data in Oracle IMDB Cache and the BRM database being unsynchronized and might result in propagation errors. In an Oracle IMDB Cache-enabled system, you must use the **pin_purge** utility to purge expired account subbalances.

See "[pin_purge](#)" for more information on how to run this utility.

How Objects Are Stored in Oracle IMDB Cache-Enabled Systems

BRM storable objects in an Oracle IMDB Cache-enabled system are assigned one of the following categories:

- *Transient objects* contain ongoing session data necessary for processing authorization and reauthorization requests that require low latency and high throughput. These objects are transactional and are stored *only* in the Oracle IMDB Cache data file. They are created and updated during a session when the system is running and deleted at the end of the session after the data is committed to the BRM database. These objects can be accessed only through simple queries.
- *Reference objects* contain data such as subscriber information and resource balances that are read often but not updated often. These objects require low latency for read-only access and are updated only at the end of the session. Reference objects are created, updated, and stored in Oracle IMDB Cache and replicated to the BRM database.
- *Database objects* are of the following types:
 - *Database-only objects* are created, updated, and stored *only* in the BRM database. Examples of these database objects include configuration objects, device objects, and pricing objects, such as products, plans, and deals.
 - *Hybrid database objects* are created and updated in Oracle IMDB Cache and propagated to the BRM database. These objects are deleted from Oracle IMDB Cache using the LRU aging policy.

About the Residency Type and Oracle IMDB Data Manager

The storable object attribute, `RESIDENCY_TYPE`, defines where the object resides in the BRM system. For information on `RESIDENCY_TYPE`, see the discussion on about the residency type in *BRM Concepts*.

IMDB Cache DM uses the residency type values to determine to which data source to send request operations. Because IMDB Cache DM uses dual connections, if the data resides in Oracle IMDB Cache, IMDB Cache DM uses the Oracle IMDB Cache connection to retrieve the data. If the data resides in the BRM database, IMDB Cache DM uses the Oracle database connection to retrieve the data.

About Storing Usage Events

By default, IMDB Cache Manager stores usage event data in expanded format. Only event objects in the `EVENT_T` base table and the `EVENT_BAL_IMPACTS_T` subtable are stored in expanded format. There are no limitations on the types of searches or stored procedures that you can use with these tables.

Using Oracle Functions or Procedures in an SQL Statement

Oracle IMDB Cache Manager does not support using Oracle functions or stored procedures in an SQL statement. To work around this limitation, trigger two separate calls to call the stored procedure and the SQL statement in your policy code.

About Searching for Usage Events in Oracle IMDB Cache-Enabled Systems

In Oracle IMDB Cache-enabled systems, usage events are created in Oracle IMDB Cache and are later propagated to the BRM database by the Oracle IMDB Cache synchronization process. To maintain memory space, Oracle IMDB Cache periodically deletes usage events from memory by using the LRU policy. Hence, at any given time, some usage events are in Oracle IMDB Cache, some are in the BRM database, and some are in both.

To ensure it finds events stored in both Oracle IMDB Cache and the BRM database, BRM performs a *union search*. Union searches consist of the following steps, which are different for each rating method:

- **Real-time rating:** BRM finds usage events by searching both Oracle IMDB Cache and the BRM database, merging the results together, and then deleting any duplicate events from the search results.
BRM uses this method to search for events only when:
 - The PCM_OP_TCF_AAA_SEARCH_SESSION helper opcode searches for duplicate session objects.
 - The PCM_OP_TCF_AAA_REFUND opcode searches for session objects to refund.
 - The PCM_OP_INV_MAKE_INVOICE opcode searches for events when creating detailed invoices.
 - The PCM_OP_BILL_MAKE_BILL_NOW opcode searches for events to bill.
- **Batch rating:** BRM finds usage events by searching the BRM database only. Before searching the database, however, BRM ensures that all usage events have been propagated from Oracle IMDB Cache to the BRM database by doing the following:
 - Retrieving a list of the most recently created and modified usage events from Oracle IMDB Cache.
 - Polling the BRM database until all the usage events in the list are propagated to the BRM database.

BRM uses this method to search for events only during the rerating process and during deferred taxation.

For information on configuring BRM to perform union searches during both batch rating and real-time rating, see ["Searching for Events in Both Oracle IMDB Cache and the BRM Database"](#).

About Committing Transactions in Both Oracle IMDB Cache and the BRM Database

Certain operations in the BRM system require updates to data in Oracle IMDB Cache and to objects in the BRM database. The *transactional consistency feature* ensures that if an error occurs while data is written to either Oracle IMDB Cache or the BRM database, the entire BRM transaction is rolled back on both data sources.

The transaction consistency feature is implemented with the following Oracle database features:

- Distributed transactions.
- Two-phase commit protocols.

- Ability to disassociate a service context from one global transaction and reassociate it with another global transaction.

Note: Any external application that works directly on both data sources without going through IMDB Cache DM must use their own consistency logic.

To configure your system to use the transaction consistency feature, see "[Configuring IMDB Cache Manager for Transaction Consistency](#)".

About Recovering from Oracle Global Transaction Errors

When an application, such as Siebel CRM, attempts to access data in Oracle IMDB Cache, IMDB Cache DM opens an Oracle Global Transaction. If the external application fails while the transaction is open, the Oracle Global Transaction performs one of the following:

- If the transaction has not been precommitted, the Oracle Global Transaction rolls back the entire transaction.
- If the transaction is precommitted, the Oracle Global Transaction keeps the transaction in an **Active** status for a specified length of time. This gives the application time to reestablish a connection to the IMDB Cache DM and complete the transaction. If the transaction is postcommitted before the wait time expires, the Oracle Global Transaction commits the transaction. If the transaction is not postcommitted before the wait time expires, the Oracle Global Transaction changes the transaction to a **Pending** status and the Oracle Database Administrator must intervene.

About Manually Fixing Oracle Global Transaction Failures

Note: For information about manually committing or rolling back XA transactions, see "Manually Committing or Rolling Back XA Transactions" in *BRM JCA Resource Adapter*.

IMDB Cache DM automatically fixes most errors that occur during the BRM transaction process. However, in rare cases, a BRM transaction error may occur that IMDB Cache DM cannot fix automatically. When this occurs, the Oracle Global Transaction is changed from an **Active** status to a **Pending** status and the error must be fixed manually.

Your Oracle Database Administrator must periodically check for and fix Oracle Global Transaction failures. [Table 20–1](#) shows the BRM tables and views that your Oracle Database Administrator should monitor.

Table 20–1 BRM Tables and Views to Monitor

Table Name	Description
GLOBAL_TRANS_T	Lists information about each Active global transaction. This IMDB Cache Manager table is stored in the data store of each logical partition.
GLOBAL_PENDING_TRANS_T	Lists the global transactions that are Pending and require manual intervention by the Oracle Database Administrator. This table is stored in your BRM database. The IMDB Cache Manager installer creates one instance of this table for each schema in your BRM database.
DBA_2PC_PENDING	Stores detailed information about each global transaction that is Pending , such as the global and local transaction ID and the state of the commit transaction. To access this Oracle static data dictionary view, you must have a SYS or SYSTEM password for the BRM database.

For more information about:

- The global transaction tables, see ["About the Global Transaction System Tables and Views"](#).
- Finding and fixing global transaction errors, see ["Finding and Fixing Global Transaction Errors"](#).

Using the Oracle IMDB Cache Grid to Partition Data

An Oracle IMDB Cache Grid is a collection of Oracle IMDB Caches that work together to cache data from the Oracle database. Cached data is distributed between the cache grid members. The cache grid manages information about the data location and provides applications access to the cached data in all the grid members with location transparency. Logical partitioning is a mechanism that provides scalability by using the Oracle IMDB Cache Grid architecture. You can configure cache grids to increase system performance and manage system growth.

Typically in a BRM system, data is distributed among multiple database schemas. You can improve scalability by distributing BRM data among multiple logical partitions in each database schema. In this configuration, data from a single database schema is distributed among multiple logical partitions (Oracle IMDB Caches). Logical partitioning essentially provides a second level of data partitioning. As your subscriber base grows, you can add additional database schemas and logical partitions. For example, for a single-schema database with 1,000,000 subscribers, you can create two logical partitions and distribute 500,000 subscriber accounts in each partition. When the subscriber base grows to 2,000,000, you can add a database schema with two more logical partitions to distribute the additional 1,000,000 new subscriber accounts.

About Logical Partitioning in High Availability Systems

In a high-availability system, a cache grid consists of one or more high-availability nodes (a pair of an active and a standby Oracle IMDB Cache). Data in the schema is partitioned among the Oracle IMDB Cache nodes based on the ACCOUNT_OBJ_DB column in the UNIQUENESS_T table.

Accounts and all associated objects reside in the same node. Each node owns the specific data partition. If one node is down, clients can continue to access accounts in

the other nodes. Because database operations are performed at the nodes where the data resides, this reduces the workload on the BRM database and improves the overall system efficiency.

In a BRM system with high availability and logical partitions, each IMDB Cache DM handles operations for a specific logical partition and database schema. All logical partitions are assigned a database number. The following naming scheme is used to denote the logical partition for a specific database schema:

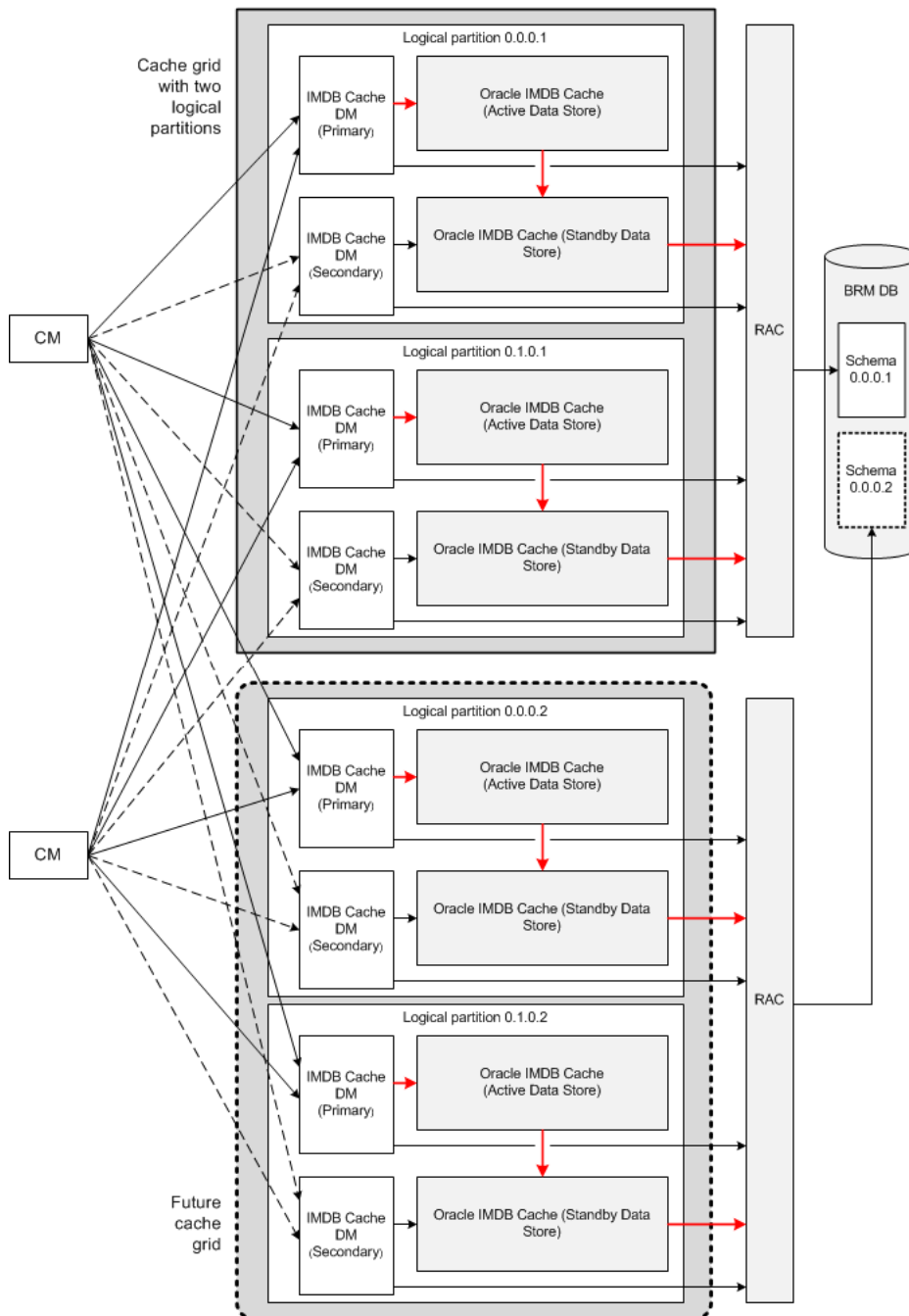
0.M.0.N

where:

- *M* specifies the logical partition.
- *N* specifies the database schema.

Note: The first logical partition is 0, not 1. For example, for database schema 0.0.0.1 (schema 1), the first partition is 0.0.0.1 and the second partition is 0.1.0.1. For database schema 0.0.0.2 (schema 2), the first partition is 0.0.0.2 and the second partition is 0.1.0.2.

[Figure 20–1](#) shows an Oracle IMDB Cache Grid architecture for a single-schema database with two logical partitions and how you can configure additional database schemas and logical partitions as your subscriber base grows.

Figure 20–1 A High-Availability Configuration with Logical Partitioning

How Accounts Are Assigned to the Logical Partition

Each database schema and logical partition is assigned a status and a priority. The status determines whether the database schema and the logical partition are available for account creation. The priorities are used during account creation to determine in which schema and partition to create the account.

BRM finds an open database schema with the highest priority and assigns accounts to an open logical partition with highest priority for that database schema. If all database schemas and logical partitions have the same priority, BRM chooses an open database schema and logical partition at random in which to create the account.

For hierarchical, brand, or group accounts, all members are created in the same database schema and generally in the same logical partition as the parent. However, it is possible that members could be located in different logical partitions based on the logical partition priority.

The database schema and logical partition status can be set to open, closed, or unavailable:

- **Open:** Database schemas and logical partitions with open status are available for account creation.
- **Closed:** Database schemas and logical partitions with closed status are not used for account creation under most circumstances. Accounts are created in a closed database schema or logical partition only if an account's parent, branded, or sponsoring account belongs to that database schema or logical partition or if all database schemas and logical partitions are closed. If all database schemas and logical partitions are closed, BRM chooses a closed database schema and logical partition at random in which to create accounts and continues to create accounts in that database schema until a database schema becomes open. To limit the number of accounts created in a database schema or logical partition, you can manually change the status to closed, or you can have BRM automatically change it to closed when the database schema or logical partition reaches a predefined limit.
- **Unavailable:** Database schemas and logical partitions with unavailable status are not used for account creation unless the database schema or logical partition contains an account's parent, sponsoring, or branded account.

To change the status or priority for a database schema or logical partition, edit the STATUS and PRIORITY entries in the *BRM_home/apps/multi_db/config_dist.conf* file, and then use the **load_config_dist** utility to load the distribution information into the primary database.

About Finding Data in Logical Partitions

In Oracle IMDB Cache-enabled systems with logical partitions, data is spread across multiple logical partitions in a Cache Grid (or schema). BRM uses the TimesTen Global Processing feature to perform the following:

- Search for data in multiple logical partitions. See ["About Performing Searches in Logical Partitions"](#).
- Retrieve data from multiple logical partitions. See ["About Retrieving Data from Logical Partitions"](#).

About Performing Searches in Logical Partitions

BRM applications and external applications can initiate the following types of searches:

- Search a single schema and return all the results simultaneously by using the PCM_OP_SEARCH opcode.
- Search a single schema and return the results in discrete chunks by using the PCM_OP_STEP_SEARCH opcode.
- Search multiple schemas and return all the results simultaneously by using the PCM_OP_GLOBAL_SEARCH opcode.
- Search multiple schemas and return the results in discrete chunks by using the PCM_OP_GLOBAL_STEP_SEARCH opcode.

When your schemas contain logical partitions, BRM can perform either a search on a specified logical partition or a global search across all logical partitions in the schema. BRM determines whether to search one or all logical partitions in the schema by reading flags passed in the call to the search opcode. [Table 20–2](#) shows the type of search the opcode performs based on the opcode call type and flag.

Table 20–2 Search Performed Based on Opcode Call Types and Flags

Opcode Call Type	Opcode Flag
Opcode called inside a transaction	<ul style="list-style-type: none"> ▪ No flag: The opcode searches the logical partition specified in the input flist. This is the default. ▪ PCM_OPFLG_SEARCH_PARTITIONS: The opcode performs a global search across all logical partitions.
Opcode called outside a transaction	<ul style="list-style-type: none"> ▪ No flag: The opcode performs a global search across all logical partitions. This is the default. ▪ PCM_OPFLG_SEARCH_ONE_PARTITION: The opcode searches the logical partition specified in the input flist.

BRM finds data in logical partitions as follows:

1. The CM routes search operations to an IMDB Cache DM instance based on the search opcode's PIN_FLD_POID input flist field. For example, if PIN_FLD_POID is set to **0.1.0.1**, the CM routes the search operation to the IMDB Cache DM instance connected to logical partition 0.1.0.1.
2. IMDB Cache DM calls the specified search opcode.
3. The search opcode executes the search against one or more logical partitions:
 - If the opcode call occurs *inside* a transaction, the opcode, by default, searches only the current logical partition. If the PCM_OPFLG_SEARCH_PARTITIONS flag is passed in the opcode call, the opcode performs a search across all logical partitions in the Cache Grid.
 - If the opcode call occurs *outside* a transaction, the opcode, by default, performs a search across all logical partitions in the Cache Grid. If the PCM_OPFLG_SEARCH_ONE_PARTITION flag is passed in the opcode call, the opcode searches only the specified logical partition.

Global processing searches can decrease search performance. Therefore, all BRM client applications, such as Customer Center and Developer Center, use global processing searches only when the required data is spread across multiple logical partitions.

If your BRM system includes an external application, you must customize the external application to use global processing searches only when absolutely necessary. For more information, see ["Searching for Data Across Logical Partitions"](#).

About Retrieving Data from Logical Partitions

BRM applications and external applications can retrieve the following types of data:

- An entire object by using the PCM_OP_READ_OBJ opcode.
- Specified object fields by using the PCM_OP_READ_FLDS opcode.

By default, the read opcodes retrieve data from only one logical partition. However, if the optional PCM_OPFLG_SEARCH_PARTITIONS flag is passed in the call to the read opcode, the opcode uses the global processing feature to retrieve data from all logical partitions in the schema.

If your BRM system includes an external application, you must customize the external application to retrieve data from multiple logical partitions. For more information, see ["Searching for Data Across Logical Partitions"](#).

How IMDB Cache DM Connects to Oracle IMDB Cache

IMDB Cache DM uses a direct driver connection to connect to Oracle IMDB Cache. For a direct driver connection, you must configure IMDB Cache DM and Oracle IMDB Cache on the same server machine.

See ["Connecting IMDB Cache DM to Your Data Stores"](#) for more information.

For more information about the direct driver connection mode, see *Oracle In-Memory Database Introduction*.

About the AAA Flow in a BRM System with IMDB Cache Manager

In an Oracle IMDB Cache-enabled BRM system, the authentication, authorization, and accounting (AAA) flow uses the **/uniqueness** table for account and service lookups to authorize accounts for prepaid services. The **/uniqueness** object contains the data required for BRM to find the account and service objects corresponding to the given login.

The PCM_OP_ACT_FIND opcode uses the **/uniqueness** object to obtain **/account** and **/service** data for a given account login. The Services Framework AAA opcodes use the data in the **/uniqueness** table to authorize and reauthorize prepaid sessions and to end prepaid accounting sessions. The Customer FM standard opcodes update the **/uniqueness** table when a new customer account is created or when a customer purchases a new service.

Installing IMDB Cache Manager

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) In-Memory Database (IMDB) Cache Manager.

Important: If you have existing BRM data created on a BRM system before IMDB Cache Manager was installed, you must run the **load_pin_uniqueness** utility to prepare data for migration to an IMDB Cache Manager-enabled system before you load BRM objects into Oracle IMDB Cache.

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

- IMDB Cache Manager. See ["Using Oracle IMDB Cache Manager"](#).
- Oracle In-Memory Database Cache concepts and architecture. See *Oracle In-Memory Database Cache User's Guide*.

Important: IMDB Cache Manager is an optional feature that requires a separate license.

About Setting Up a BRM System with IMDB Cache Manager

Before you set up a BRM system with IMDB Cache Manager, you should plan your system configuration. For example, you should decide whether the system will be configured with logical partitioning, high availability, a single-schema database, or a multischema database.

About Setting Up IMDB Cache Manager for a Basic BRM System

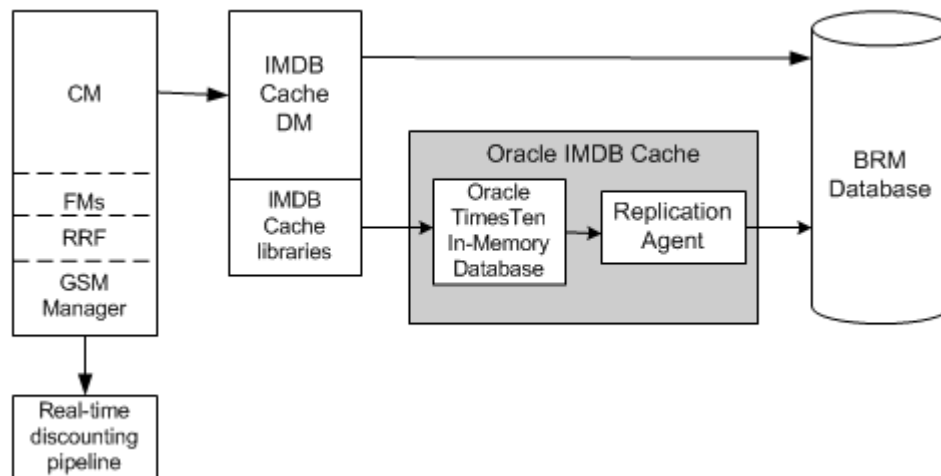
A basic BRM system for rating prepaid and postpaid real-time events using IMDB Cache Manager includes the following components:

- Client applications (for example, Customer Center)
- Service-specific applications (for example, GSM AAA Manager and GPRS AAA Manager)
- Connection Manager, along with the Facilities Modules (FMs)
- Resource Reservation Framework (RRF)
- Real-time discounting pipeline, if your rating includes discounting
- IMDB Cache DM

- Oracle IMDB Cache
- Oracle Database

Figure 21–1 shows a basic BRM system for rating prepaid real-time events with IMDB Cache Manager.

Figure 21–1 A Basic BRM System with IMDB Cache Manager



About Setting Up IMDB Cache Manager in a Multischema System

In a BRM system with multiple database schemas, each schema must be mapped to an instance of IMDB Cache DM. Each IMDB Cache DM handles all the operations associated with the accounts in that specific schema. You can configure one CM for each IMDB Cache DM, one CM for multiple IMDB Cache DMs, or multiple CMs for multiple IMDB Cache DMs.

In a BRM system with multiple database schemas, each Oracle IMDB Cache instance is associated with one schema. There is no overlap in the data stored in the different instances of Oracle IMDB Cache.

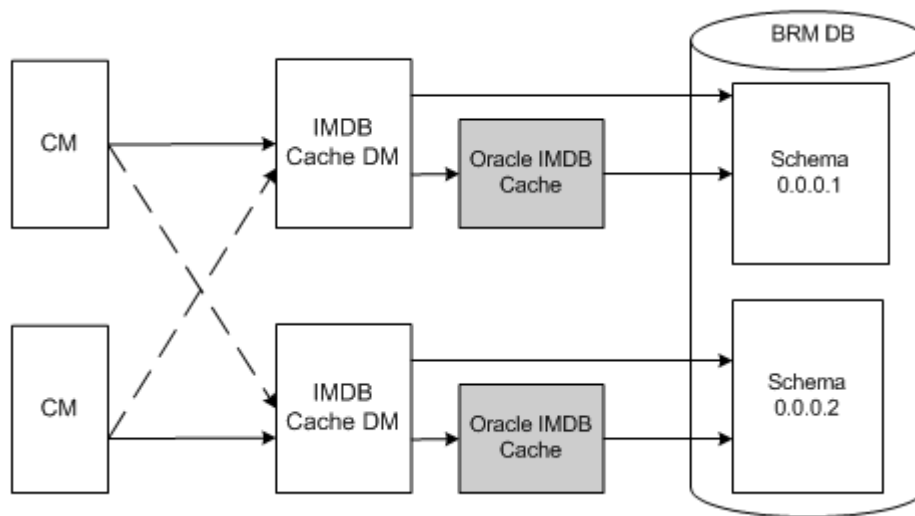
The installation and configuration steps for setting up IMDB Cache Manager for a multischema system are the same as for a single-schema system. For each schema, follow the steps in ["Installing and Configuring a BRM System with IMDB Cache Manager"](#) to install and configure IMDB Cache Manager and Oracle IMDB Cache, except for the following:

- Before you run **pin_tt_schema_gen -a** to generate the schema and load SQL files, the following entries must be changed in *BRM_home/bin/pin_tt_schema_gen.values* file for the specific schema:
 - Update the schema SQL file name to refer to the database number for the schema. For example:
`$fileName="tt_schema_0.0.0.2.sql";`
 - Update the database user name for the schema. For example:
`$MAIN_DB{ 'user' }="Secondary_Schema_User";`
 - Update the password for the schema database user. For example:
`$MAIN_DB{ 'password' }="Secondary_Schema_User_Password";`

- Specify the user name of the primary schema. For example:
`$primary_db_user_name="Primary_Schema_User";`
- Update the database number for the logical partition. For example:
`$db_no_for_load_sql="0.0.0.2";`
- During IMDB Cache Manager installation, provide the database number for the specific schema. For example, the database number for the second schema should be 0.0.0.2.

Figure 21–2 shows a basic IMDB Cache-enabled system for multiple database schemas.

Figure 21–2 Basic IMDB Cache-Enabled System for a Multischema Database



Hardware and Software Requirements

Table 21–1 describes the hardware and software requirements for installing IMDB Cache Manager.

Table 21–1 Hardware and Software Requirements for IMDB Cache Manager

System Requirement	Hardware and Software Version
Operating System	Solaris and Linux
Processor	64-bit architecture
Database Version	<ul style="list-style-type: none"> ■ Oracle 11.1.0.7.0 and required Oracle patches 6890831, 9352179, 6963600. ■ Oracle 11.2.0.1.0
Oracle IMDB Cache	Oracle IMDB Cache 11.2.1.8.2 Oracle Clusterware 11.1.0.7.0 Run Oracle IMDB Cache in its own cluster, not in the Oracle Real Application Clusters (Oracle RAC) cluster.
Java Runtime	J2SE Software Development Kit (JDK) version 1.5

Table 21–1 (Cont.) Hardware and Software Requirements for IMDB Cache Manager

System Requirement	Hardware and Software Version
Perl	Perl 5.8.0 Perl Modules: <ul style="list-style-type: none"> ▪ DBI version 1.605 ▪ DBD-Oracle version 1.16 ▪ Bit-Vector version 7.1

Installing and Configuring a BRM System with IMDB Cache Manager

This section provides instructions for configuring a basic BRM system with IMDB Cache Manager. The examples demonstrate how to create the data store **tt_0.0.0.1** in Oracle IMDB Cache for a single-schema BRM database.

To install and configure a basic BRM system with IMDB Cache Manager:

1. Install and configure the BRM database.
2. Install Oracle IMDB Cache. See *Oracle TimesTen In-Memory Database Installation Guide*.
3. Install BRM 7.5.
4. Install IMDB Cache Manager. See ["Installing IMDB Cache Manager"](#).
5. Install any optional components that you want to add to your system.
6. Create your data stores. See ["Creating the Data Store in Oracle IMDB Cache"](#).
7. If you have existing BRM data that was created before IMDB Cache Manager was installed, run the **load_pin_uniqueness** utility to prepare data for migration to an IMDB Cache-enabled system.

Note: Stop and restart CM and DM.

8. Extract the data you want to cache from the BRM database and configure your cache groups schema. See ["Generating the BRM Cache Group Schema"](#).
9. Initialize your data stores. See ["Initializing Your Data Stores in Oracle IMDB Cache"](#).
10. Connecting your data stores to the BRM database. See ["Connecting Your Data Stores to the BRM Database"](#).
11. Connect IMDB Cache DM to your data stores. See ["Connecting IMDB Cache DM to Your Data Stores"](#).
12. Configure your CM to connect to the IMDB Cache DM. See ["Connecting the CM to IMDB Cache DM"](#).

Installing IMDB Cache Manager

Note: The base BRM schema is updated when you install IMDB Cache Manager. The schema updates cannot be undone.

To install IMDB Cache Manager, perform these tasks:

- [Obtaining Information Needed for Installing IMDB Cache Manager](#)
- [Installing IMDB Cache Manager](#)
- [Running the pin_setup Script](#)
- [Granting Privileges to the Global Transaction Tables](#)

Obtaining Information Needed for Installing IMDB Cache Manager

You will be prompted for the following information about your existing BRM system during the IMDB Cache Manager installation:

Tip: You can obtain some of the information from the existing *BRM_home/setup/pin_setup.values* file on your BRM server.

- The temporary directory for installation (*temp_dir*)
- The following information about the BRM database to which IMDB Cache DM will connect:
 - Database alias
 - Database user name
 - Database user password
- The following information about the machine running the Oracle IMDB Cache:
 - Data store name
 - Database number
 - Port number

Installing IMDB Cache Manager

Important: IMDB Cache Manager must be installed on the same system where Oracle IMDB Cache is installed.

To install IMDB Cache Manager:

1. Go to the Oracle Software Delivery Cloud web site (<http://edelivery.oracle.com>).
2. Download the software to a temporary directory (*temp_dir*).

Important:

- If you download to a Windows workstation, use **FTP** to copy the **.bin** file to a temporary directory on your UNIX server.
 - You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid “Out of Memory” error messages in the log file.
-
-

3. Go to the directory where you installed the Third-Party package and source the **source.me** file.

Caution: You must source the **source.me** file to proceed with installation, otherwise “suitable JVM not found” and other error messages appear.

Bash shell:

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

C shell:

```
source source.me.csh
```

4. Go to *temp_dir* and enter this command:

```
7.5.0_TimesTen_Manager_platform.bin
```

Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the **DISPLAY** environment variable before you install the software.

5. Follow the instructions on the screen and, when prompted, provide the information you collected. The default installation directory for IMDB Cache Manager is **opt/portal/7.5**.

Note: If you plan to install IMDB Cache Manager with high availability, ensure that you enter the following details during installation:

```
Restrict access to the TimesTen installation to the group 'pin'? [
yes ] no
Do you want to restrict access to the TimesTen installation to a
different group? [ yes ] no
Are you sure you want to make this instance world-accessible? [ yes
] yes
```

6. Go to the directory where you installed the IMDB Cache Manager package and source the **source.me** file:

Bash shell:

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

C shell:

```
source source.me.csh
```

Running the **pin_setup** Script

The **pin_setup** script reads the **pin_setup.values** file, configures IMDB Cache DM, and starts IMDB Cache DM. If necessary, open the *BRM_home/setup/pin_setup.values* file and change the configuration entries.

To run the **pin_setup** script:

1. Log in as user **pin**.
2. Go to the *BRM_home/setup* directory and run the **pin_setup** script:

```
./pin_setup
```
3. Check the *BRM_home/setup/pin_setup.log* file for status and errors.

Granting Privileges to the Global Transaction Tables

To grant read privileges to the IMDB Cache global transaction tables, perform the following for *each* BRM database user:

1. Using SQL*Plus, log in to your database as the SYSTEM user.

```
sqlplus system/manager@databaseAlias
```
2. Grant read privileges to the global transaction tables:

```
GRANT SELECT ON SYS.DBA_2PC_PENDING TO User;  
GRANT SELECT ON SYS.DBA_2PC_NEIGHBORS TO User;
```
3. Exit SQL*Plus.
4. To initialize the database instance with your changes, stop and restart the BRM database.

Creating the Data Store in Oracle IMDB Cache

You create data stores in Oracle IMDB Cache for storing your BRM database tables. When creating your data stores, ensure that they adhere to the following naming convention:

tt_0.M.0.N

where:

- *M* specifies the logical partition. The first logical partition is 0, the second logical partition is 1, and so on.
- *N* specifies the database schema.

Creating Data Stores for a Basic BRM System

To create a data store in Oracle IMDB Cache:

1. On the system where Oracle IMDB Cache is installed, create a directory for storing database files:

```
mkdir IMDB_home/Database_Files_Location
```

where:

- *IMDB_home* is the directory in which you installed Oracle IMDB Cache.
- *Database_Files_Location* is the directory in which you want to store the BRM database files.

For example:

```
mkdir IMDB_home/BRMFiles
```

Note: Oracle recommends using a local disk for the database files instead of a network-mounted disk.

2. Go to the user home directory:

```
cd IMDB_home/info
```

3. Create a data store configuration file named **sys.odbci.ini** with the following entries:

Note: For information on configuring the data store attributes in the **sys.odbci.ini** file, see *Oracle TimesTen In-Memory Database Operations Guide*.

sys.odbci.ini configuration file

```
[DSN]
DataStore=Database_Files_Location/Datastore_Name
OracleNetServiceName=Oracle_Database_Service_Name
DatabaseCharacterSet=Character_Set
ConnectionCharacterSet=Character_Set
PLSQL=1
OracleNetServiceName=BRM_Database_Service_Name
oraclepwd=Oracle_DB_Password
Driver=$TIMESTEN_HOME/lib/libtten.so
#Shared-memory size in megabytes allocated for the datastore.
PermSize= Shared_Memory_Size
#Shared-memory size in megabytes allocated for temporary data partition,
generally half the size of PermSize.
TempSize=Half_of_PermSize
PassThrough=0
#Use large log buffer, log file sizes
LogFileSize=512
#Async repl flushes to disk before sending batches so this makes it faster
#on Linux
LogFlushMethod=2
CkptFrequency=200
CkptLogVolume=0
#Limit Ckpt rate to 10 mb/s
CkptRate=10
Connections=200
#Oracle recommends setting LockWait to 30 seconds.
LockWait=30
DurableCommits=0
CacheGridEnable=1
```

where:

- *DSN* is the data source name, which is the same as the data store name. DSN must also be the same as the database alias name in the **tnsnames.ora** file.
- *DatabaseFilesLocation* is the directory where you want to store the database files.
- *DataStoreName* is the name of the data store in the Oracle IMDB Cache.
- *Oracle_Database_Service_Name* identifies the service name for the BRM database instance.
- *Character_Set* specifies the database character set of either **UTF8** or **AL32UTF8**. This value must match the BRM database character set.
- *Oracle_DB_User* is the BRM database user.

- *Oracle_DB_Password* is the password for the BRM database user.
 - *Shared_Memory_Size* is the size of the data store's shared-memory segment.
4. Save and close the file.
 5. Go to *IMDB_home/bin* and source the **ttenv.csh** file:

```
cd IMDB_home/bin
source ttenv.csh
```

6. Set up the Oracle IMDB Cache grid privileges in the BRM database:
 - a. Connect to the BRM database as a system administrator:

```
cd IMDB_home/oraclescripts
sqlplus sys as sysdba
```

- b. Run the following SQL scripts:

```
@IMDB_home/oraclescripts/initCacheGlobalSchema.sql
@IMDB_home/oraclescripts/grantCacheAdminPrivileges.sql
```

- c. Run the following commands to grant privileges:

```
grant all on TIMESTEN.TT_GRIDID to "Oracle_DB_User";
```

where *Oracle_DB_User* is the BRM database user.

For more information, see *Oracle In-Memory Database Cache User's Guide*.

7. Create the data store by using the TimesTen **ttIsql** utility:

```
cd IMDB_home/bin
ttIsql DataStoreName
```

where *DataStoreName* is the name of the data store you defined in the **sys.odbc.ini** file.

8. Create the data store user:

```
create user DataStoreName identified by DataStorePassword;
```

where:

DataStoreUser is the Oracle IMDB Cache data store user.

DataStorePassword is the password for the Oracle IMDB Cache data store user.

Important: The Oracle IMDB Cache data store user must be the same as the BRM database user. However, the data store password can be different from the BRM database user password.

To initialize data stores for a multischema setup, you must create the primary user along with the secondary user in all the secondary logical partitions.

```
create user Primary_Data_Store_User identified by Primary_Data_Store_Password;
```

where:

- *Primary_Data_Store_User* is the primary data store user.
- *Primary_Data_Store_Password* is the password for the primary data store user.

9. Grant all permissions to the data store user:

```
grant all to Data_Store_User;
```

10. Set the data store user and password:

```
ttIsql "uid=Data_Store_User; pwd=Data_Store_Password; dsn=tt_0.0.0.1"
call ttcacheuidpwdset('Cacheadminuser','Cacheadminuserpwd');
```

where:

- *Cacheadminuser* is the cache user name.
- *Cacheadminuserpwd* is the cache password.

11. Make the data store grid-enabled.

For example:

```
call ttGridCreate('ttGrid');
call ttGridNameSet('ttGrid');
call ttGridAttach(1,'LogicalPartition',Host, Port);
```

where:

- *LogicalPartition* is the logical partition number, such as 0.1.0.1.
- *Host* is the host on which the data store resides.
- *Port* is a free port on the system.

See *Oracle TimesTen In-Memory Database Reference* for more information on the procedures for creating a data store.

Creating Data Stores for Logical Partitioning

To create the cache grid **ttGrid** with data stores **tt_0.0.0.1** and **tt_0.1.0.1**:

1. On the system where Oracle IMDB Cache is installed, create a directory for storing database files:

```
mkdir BRM_home/Database_Files_Location
```

For example:

```
mkdir BRM_home/brm_database_files
```

Note: Oracle recommends using a local disk for database files instead of a network mounted disk.

2. Go to the user home directory:

```
cd IMDB_home/info
```

3. Create (or edit) a data store configuration file named **sys.odbc.ini** with the data store attributes.

For example:

```
[tt_0.0.0.1]
DataStore=/opt/portal/7.5/brm_database_files/tt_0.0.0.1
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
PLSQL=1
OracleNetServiceName=pin_db
oraclepwd=pin01
```

```

Driver=$TIMESTEN_HOME/lib/libtten.so
# Shared-memory size in megabytes allocated for the datastore.
PermSize=32
#Shared-memory size in megabytes allocated for temporary data partition,
generally half the size of PermSize.
TempSize=16
PassThrough=0
#Use large log buffer, log file sizes
LogFileSize=512
#Async repl flushes to disk before sending batches so this makes it faster
#on Linux
LogFlushMethod=2
#Limit Ckpt rate to 10 mb/s
CkptFrequency=200
CkptLogVolume=0
CkptRate=10
Connections=200
#Oracle recommends setting LockWait to 30 seconds.
LockWait=30
DurableCommits=0
CacheGridEnable=1

[tt_0.1.0.1]
DataStore=/opt/portal/7.5/brm_database_files/tt_0.1.0.1
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
PLSQL=1
OracleNetServiceName=pin_db
oraclepwd=pin01
Driver=$TIMESTEN_HOME/lib/libtten.so
#The shared-memory size in megabytes allocated for the datastore.
PermSize=32
#Shared-memory size in megabytes allocated for temporary data partition,
generally half the size of PermSize.
TempSize=16
PassThrough=0
#Use large log buffer, log file sizes
LogFileSize=512
#Async repl flushes to disk before sending batches so this makes it faster
#on Linux
LogFlushMethod=2
#Limit Ckpt rate to 10 mb/s
CkptFrequency=200
CkptLogVolume=0
CkptRate=10
Connections=200
#Oracle recommends setting LockWait to 30 seconds.
LockWait=30
DurableCommits=0
CacheGridEnable=1

```

Note: In the examples above [tt_0.0.0.1] is the DSN for data store tt_0.0.0.1 and [tt_0.1.0.1] is the DSN for data store tt_0.1.0.1. The DSNs must be the same as the database alias name in the **tnsnames.ora** file.

4. Save and close the file.
5. Go to the directory where you installed Oracle IMDB Cache and source the **ttenv.csh** file:

```
cd IMDB_home/bin
source ttenv.csh
```

6. Set up the Oracle IMDB Cache grid privileges in the BRM database:

- a. Connect to the BRM database as a system administrator:

```
cd IMDB_home/oraclescripts
sqlplus sys as sysdba
```

- b. Run the following SQL scripts:

```
IMDB_home/oraclescripts/initCacheGlobalSchema.sql
IMDB_home/oraclescripts/grantCacheAdminPrivileges.sql
```

- c. Run the following commands:

```
grant all on TIMESTEN.TT_GRIDID to "Oracle_DB_User";
grant all on TIMESTEN.TT_GRIDINFO to "Oracle_DB_User";
```

where *Oracle_DB_User* is the BRM database user.

For more information, see *Oracle TimesTen In-Memory Database Cache User's Guide*.

7. Perform the following tasks on the Oracle IMDB Cache system.

- a. Create the data stores for each logical partition. For example:

```
cd IMDB_home/bin
ttIsql tt_0.0.0.1
ttIsql tt_0.1.0.1
```

- b. Create the data store user for each data store and grant all permissions:

```
ttIsql tt_0.0.0.1
create user TimesTen_DB_User identified by TimesTen_DB_Password;
grant all to TimesTen_DB_User;
```

```
ttIsql tt_0.1.0.1
create user TimesTen_DB_User identified by TimesTen_DB_Password;
grant all to TimesTen_DB_User;
```

where:

TimesTen_DB_User is the Oracle IMDB Cache data store user and must be the same as the BRM database user.

TimesTen_DB_Password is the Oracle IMDB Cache data store password.

Important: *TimesTen_DB_User* must be the same as the BRM database user.

TimesTen_DB_Password can be different from the BRM database user password.

- c. Set the data store user and password and make each data store grid-enabled:

```
ttIsql "uid=TimesTen_DB_User;pwd=TimesTen_DB_Password;dsn=tt_0.0.0.1"
call ttcacheuidpwdset('Cache_Admin_User', 'Cache_Admin_User_Pwd');
call ttGridCreate('ttGrid');
call ttGridNameSet('ttGrid');
```

```
ttIsql "uid=TimesTen_DB_User;pwd=TimesTen_DB_Password;dsn=tt_0.1.0.1"
```



```
call ttcacheuidpwdset('Cache_Admin_User', 'Cache_Admin_User_Pwd');
call ttGridNameSet('ttGrid');
```

where:

Cache_Admin_User is the cache user.

Cache_Admin_User_Pwd is the cache user password.

For more information on the procedures for creating a cache grid, see *Oracle TimesTen In-Memory Database Cache User's Guide*.

Initializing Your Data Stores in Oracle IMDB Cache

After following the instructions in ["Generating the BRM Cache Group Schema"](#) for generating your schema and load SQL scripts, you must create and initialize the schema in the Oracle IMDB Cache data store.

To initialize your data stores, perform the following steps *for each data store* in your system:

1. Set the data store user and password:

```
ttIsql "uid=TimesTen_DB_User; pwd=TimesTen_DB_Password; dsn=tt_0.0.0.1"
call ttcacheuidpwdset('Cache_Admin_User', 'Cache_Admin_User_Pwd');
```

2. Start the cache agent:

```
call ttcachestart;
```

3. Create the schema:

```
run BRM_home/bin/tt_schema.sql;
```

4. Create stored procedures:

```
run BRM_home/sys/dm_tt/data/tt_create_pkg_pin_sequence.plb;
run BRM_home/sys/dm_tt/data/tt_create_procedures.plb;
run BRM_home/sys/dm_tt/data/create_tt_wrappers.plb;
```

Note: The stored procedures in `tt_create_pkg_pin_sequence.plb` must be loaded before the procedures in `tt_create_procedures.plb`.

5. If your system uses logical partitioning, attach the data store to the cache grid:

```
call ttGridAttach(1, 'LogicalPartition', 'Host', Port);
```

where

- *LogicalPartition* is the logical partition number, such as 0.1.0.1.
- *Host* is the host on which the data store resides
- *Port* is a free port on the system

6. Load the BRM data into the data store:

```
run BRM_home/bin/tt_load.sql;
```

7. Get the status and exit:

```
call ttrepstart;
statsupdate;
```

```
exit;
```

Connecting Your Data Stores to the BRM Database

To configure your data stores to establish connections with the BRM database:

1. Open the **tnsnames.ora** file located in the directory specified by \$TNS_ADMIN.
2. Add the following entry:

Note: For systems with logical partitions, add a separate **tnsnames.ora** entry for each logical partition.

```
Database_Alias_Name = (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT =  
))  
(CONNECT_DATA = (SERVICE_NAME = DataStore_Name) (SERVER = timesten_direct )))
```

where:

- *Database_Alias_Name* is the DSN specified in the **sys.odbci.ini** file
- *DataStore_Name* is the name of the data store specified in the **sys.odbci.ini** file
- **timesten_direct** indicates that Oracle IMDB Cache can connect directly to the BRM database

For example, if your system contains two logical partitions (**0.0.0.1** and **0.1.0.1**):

```
tt_0.0.0.1= (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT = ))  
(CONNECT_DATA = (SERVICE_NAME = tt_0.0.0.1) (SERVER = timesten_direct )))
```

```
tt_0.1.0.1= (DESCRIPTION = (ADDRESS = (PROTOCOL = ) (HOST = ) (PORT = ))  
(CONNECT_DATA = (SERVICE_NAME = tt_0.1.0.1) (SERVER = timesten_direct )))
```

3. Save and close the file.

Note: To initialize data stores for a multischema setup, you must generate the schema and load SQL files for each database schema by using the **pin_tt_schema_gen** utility. Then, follow the procedures in ["Creating the Data Store in Oracle IMDB Cache"](#) and ["Initializing Your Data Stores in Oracle IMDB Cache"](#) to create and initialize the data stores for each schema.

Connecting IMDB Cache DM to Your Data Stores

Each IMDB Cache DM instance must be connected to a data store.

- For nonpartitioned systems, see ["Connecting the IMDB Cache DM to Your Data Store"](#).
- For partitioned systems, see ["Connecting Each IMDB Cache DM Instance to a Data Store"](#).

Connecting the IMDB Cache DM to Your Data Store

To configure IMDB Cache DM to connect to your data store:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.

2. Set the **sm_database_tt** entry to the data store. For example:

```
- dm sm_database_tt tt_0.0.0.1
```

3. Set the **sm_pw_tt** entry to the data store password.

```
- dm sm_pw_tt Datastore_Password
```

4. Save and close the file.

5. Restart IMDB Cache DM.

Connecting Each IMDB Cache DM Instance to a Data Store

For logical partitioning, there is a one-to-one relationship between data stores and IMDB Cache DM instances. This section describes how to create additional IMDB Cache DM instances and connect each one to a unique data store.

To connect each IMDB Cache DM instance to a data store:

- Connect your existing IMDB Cache DM instance to your first data store. See ["Connecting IMDB Cache DM to Your First Data Store"](#).
- Add a new IMDB Cache DM instance for each subsequent data store and connect each one to a data store. See ["Creating Additional IMDB Cache DM Instances for Each Subsequent Data Store"](#).

Connecting IMDB Cache DM to Your First Data Store

Use the existing IMDB Cache DM directory to configure the settings for your first data store.

To configure the IMDB Cache DM for your first data store:

1. Open the *BRM_home/sys/dm_tt/pin.conf* file in a text editor.
2. Set the **sm_database_tt** entry to the data store's SQL*Net alias that you defined in the *tnsnames.ora* file. For example, for data store **tt_0.0.0.1**:

```
- dm sm_database_tt tt_0.0.0.1
```

3. Set the **sm_tt_pw** entry to the data store password:

```
- dm sm_tt_pw Datastore_Password
```

4. Set the **logical_partition** entry to **1** to enable logical partitioning:

```
- dm logical_partition 1
```

5. Save and close the file.

Creating Additional IMDB Cache DM Instances for Each Subsequent Data Store

You add new IMDB Cache DM instances to your system by making copies of the existing IMDB Cache DM directory. You then connect each new IMDB Cache DM instance to its associated data store.

To connect a new IMDB Cache DM instance to a subsequent data store:

1. Add a new IMDB Cache DM instance by making a copy of the IMDB Cache DM directory:
 - a. In *BRM_home/sys*, create a subdirectory named **dm_tt_DataStoreNumber**. For example, create a subdirectory named **dm_tt_0.1.0.1** for data store **tt_0.1.0.1**.
 - b. Copy the contents of *BRM_home/sys/dm_tt* into **dm_tt_DataStoreNumber**.

For example, copy the contents of *BRM_home/sys/dm_tt* into **dm_tt_0.1.0.1**

2. Connect the new IMDB Cache DM instance to its associated data store:
 - a. Open the *BRM_home/sys/dm_tt/dm_tt_DataStoreNumber/pin.conf* file in a text editor.
 - b. Update the following entries:
 - dm sm_database_tt **tt_0.1.0.1**
 - dm sm_pw_tt *Datastore_Password*
 - dm logical_partition **1**
 - c. Configure **dm_tt_0.1.0.1** to have its own log file location and its own port:
 - dm dm_logfile *BRM_home/sys/dm_tt_0.1.0.1/dm_tt_0.1.0.1.pinlog*
 - dm dm_port *Port*
 - d. Save and close the file.
3. Configure the **pin_ctl** utility for starting and stopping the new IMDB Cache DM instance:

- a. Open the *BRM_home/bin/pin_ctl.conf* file in a text editor.

- b. Add the following line to the components list:

```
start_sequence DMTT_Instance2_Service_Name=DMTT_Instance1_Service_Name
```

where:

start_sequence is the start and stop sequence number. This determines the order in which components are started or stopped.

DMTT_Instance2_service_name is the name of the IMDB Cache DM instance.

DMTT_Instance1_service_name indicates that *DMTT_Instance2_service_name* is different from the standard service name.

For example:

```
1 dm_tt #This line is preconfigured with IMDB Cache DM installation.
```

```
1 dm_tt2=dm_tt #This line is added for multiple IMDB Cache DM instances.
```

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

```
start_DMTTInstance_Service_Name cpidproc:DMTTInstance_Service_Name:  
cport:DMTT_Port host:DMTT_Host dbno:DSN
```

where:

start_DMTTInstance_Service_Name is the name of the start command for the IMDB Cache DM instance.

cpidproc:DMTTInstance_Service_Name is a simple process name matching filter.

cport:DMTT_Port is the IMDB Cache DM port number.

host:DMTT_Host is the IMDB Cache DM host name.

dbno:DSN is the data store database number.

For example:

```
start_dm_tt cpidproc:dm_tt: cport:1234 host:vm31230 dbno:0.0.0.1
```

```
start_dm_tt2 cpidproc:dm_tt2: cport:2233 host:vm31230 dbno:0.1.0.1
```

d. Save and close the file.

See ["Using the pin_ctl Utility to Monitor BRM"](#).

4. Go to the *BRM_home/bin* directory and start the IMDB Cache DM instance by running the following command:

```
pin_ctl start dm_tt
```

Connecting the CM to IMDB Cache DM

To configure the Connection Manager (CM) to connect to IMDB Cache DM:

1. Open the CM **pin.conf** file (*BRM_home/sys/cm/pin.conf*) in a text editor.
2. Set the **enable_publish** entry to 0:


```
- fm_publish enable_publish 0
```
3. Add a **dm_pointer** entry for each IMDB Cache DM instance in your system. Replace *Hostname* with the host of the IMDB Cache DM instance, and replace *Port* with the port number of the IMDB Cache DM instance.

For example, if your system contains two IMDB Cache DM instances:

```
- cm dm_pointer 0.0.0.1 ip Hostname Port
- cm dm_pointer 0.1.0.1 ip Hostname Port
```

4. Add a **dm_attribute** entry for each logical partition in your system:

```
- cm dm_attributes LogicalPartition Attribute, Attribute, ...
```

where:

LogicalPartition specifies the logical partition number.

Attribute is set to one or more of the following:

- **scoped:** Use this option only when you use branding. It restricts database access for separate brands. You can disable this option if branding is not used.
- **assign_account_obj:** Assigns an owner account reference when the object is created. If you use branding, all objects are associated with an account. You can disable this option if branding is not used.
- **searchable:** Limits access to certain logical partitions in a multischema environment. The CM can search all nodes in a single cache grid or perform a global search on all nodes in all the cache grids. This option lets the CM know which nodes are searchable.

For example, if your system contains two logical partitions:

```
- cm dm_attributes 0.0.0.1 assign_account_obj, scoped, searchable
- cm dm_attributes 0.1.0.1 assign_account_obj, scoped, searchable
```

5. Set the **timesten** entry to 1, which indicates that this environment uses Oracle IMDB Cache:

```
- cm timesten 1
```

6. If your system contains logical partitions, set the **logical_partition** entry to 1:

```
-cm logical_partition 1
```

7. Save and close the file.
8. Verify that IMDB Cache DM is running.
9. Restart the CM.

About Installing Optional Components

Consider a scenario where you are using BRM with IMDB Cache Manager, and you install an additional optional component, which creates additional tables in the BRM database. You may want these additional tables to be cached in Oracle IMDB Cache to improve the performance of your BRM system.

To cache additional tables in Oracle IMDB Cache, you must rerun the **pin_tt_schema_gen** utility, schema SQL script, and load SQL script. See "[Generating the BRM Cache Group Schema](#)".

Note: You may need to configure your BRM system for using the additional component with Oracle TimesTen In-Memory Database cache.

Uninstalling IMDB Cache Manager

To uninstall IMDB Cache Manager, run the **uninstaller.bin** utility from the *BRM_home/uninstaller/TimesTen_Manager* directory.

Configuring IMDB Cache Manager

This chapter describes how to configure an Oracle Communications Billing and Revenue Management (BRM) system with In-Memory Database (IMDB) Cache Manager.

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

- IMDB Cache Manager. See ["Using Oracle IMDB Cache Manager"](#).
- Oracle IMDB Cache concepts and architecture. See *Oracle In-Memory Database Cache User's Guide*.

IMDB Cache Manager Configuration Tasks

You can configure IMDB Cache Manager in the following ways.

- [Configuring IMDB Cache Manager for Transaction Consistency](#)
- [Searching for Events in Both Oracle IMDB Cache and the BRM Database](#)
- [Customizing External Applications for Logical Partitions](#)
- [Configuring Your Event Tables for BRM Reports](#)
- [Configuring How to Store Reservations](#)
- [Setting the Stacksize Limit](#)
- [Setting Database Schema Status for Oracle IMDB Cache Systems](#)
- [Setting the Database Schema Priority for Oracle IMDB Cache Systems](#)

Configuring IMDB Cache Manager for Transaction Consistency

To configure IMDB Cache Manager to use the transaction consistency feature:

- Ensure that the transaction consistency feature is enabled. See ["Enabling the Transaction Consistency Feature"](#).
- Specify how long to keep transactions in an **Active** status after an error. See ["Specifying How Long to Keep Transactions Active after an Error"](#).

For more information, see ["About Committing Transactions in Both Oracle IMDB Cache and the BRM Database"](#).

Enabling the Transaction Consistency Feature

When you install IMDB Cache Manager, transaction consistency is automatically enabled.

- When enabled, IMDB Cache DM can perform operations on both Oracle IMDB Cache and the BRM database in a single transaction. This guarantees that the data will be consistent across both data sources.
- When disabled, BRM can perform read-write operations when the data spans both data sources. However, the data will be inconsistent if one of the transactions fail.

You specify whether transaction consistency is enabled by editing the **dm_trans_consistency_enabled** entry in the IMDB Cache DM configuration file.

Note: The active and standby IMDB Cache DM instances must have the same configuration settings in the **dm_tt pin.conf** file. In particular, the number of IMDB Cache DM back-end process must be the same.

To enable the transaction consistency feature, perform the following for each IMDB Cache DM instance:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.
2. Set the **dm_trans_consistency_enabled** entry to 1:
 - dm dm_trans_consistency_enabled 1
 - Setting this entry to 1 enables the transaction consistency feature. This is the default.
 - Setting this entry to 0 disables the transaction consistency feature.
3. Set the **dm_restart_children** entry to 1:
 - dm dm_restart_children 1
 - Setting this entry to 1 specifies to replace child processes that have stopped. This prevents the system from losing DM processes because of transient failures over time. This is the default.
 - Setting this entry to 0 specifies to not replace child processes that have failed.
4. Save and close the file.
5. Restart the IMDB Cache DM instance.

Specifying How Long to Keep Transactions Active after an Error

When an error occurs during a transaction, the Oracle Global Transaction waits, by default, for 3600 seconds (or 1 hour). You can increase or decrease the wait time by modifying the **dm_trans_timeout_in_secs** entry in the IMDB Cache DM **pin.conf** file. You should base the wait time on how long it would take your system to recover from a failure or a node switchover. The minimum wait time is 10 seconds, and the maximum wait time is 5,184,000 seconds (or 2 months).

Note: For information about the **dm_xa_trans_timeout_in_secs** entry, see "Changing the XA Transaction Timeout Period" in *BRM JCA Resource Adapter*.

To specify how long to keep Oracle Global Transactions in an **Active** status after an error, perform the following for *each* IMDB Cache DM instance:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.
2. Set the **dm_trans_timeout_in_secs** entry to the appropriate value:

```
-dm dm_trans_timeout_in_secs value
```

where *value* specifies the amount of time, in seconds, to keep the global transaction in an **Active** state after an error.

3. Save and close the file.
4. Restart the IMDB Cache DM instance.

Searching for Events in Both Oracle IMDB Cache and the BRM Database

To configure BRM to perform event searches in both Oracle IMDB Cache and the BRM database:

- Ensure that the union search feature is enabled. See ["Enabling Union Searches for Events"](#).
- Specify a timeout value for batch searches. See ["Specifying the Timeout Value for Batch Polling Operations"](#).
- If necessary, customize any external applications to perform batch polling. See ["Customizing External Applications for Real-Time Union Searches"](#).
- If necessary, customize any external applications to perform union searches during batch rating. See ["Customizing External Applications for Batch Polling Operations"](#).

For more information, see ["About Searching for Usage Events in Oracle IMDB Cache-Enabled Systems"](#).

Enabling Union Searches for Events

When the union search feature is enabled, BRM finds usage events by searching both Oracle IMDB Cache and the BRM database.

When the union search feature is disabled, BRM searches for usage events in either Oracle IMDB Cache or the BRM database, depending on the residency type.

You specify whether union search is enabled by editing the **tt_unionsearch_enabled** entry in the IMDB Cache DM configuration file.

To enable union searches, perform the following for each IMDB Cache DM instance:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.
2. Set the **tt_unionsearch_enabled** entry to 1:

```
-dm tt_unionsearch_enabled 1
```

- Setting this entry to 1 enables union searches. This is the default.
- Setting this entry to 0 disables union searches.

3. Save and close the file.
4. Restart the IMDB Cache DM instance.

Specifying the Timeout Value for Batch Polling Operations

When performing union searches during batch operations, BRM ensures that all recently created and modified usage events have been propagated from Oracle IMDB Cache to the BRM database by doing the following:

- Retrieving the list of events that must be propagated from Oracle IMDB Cache.
- Polling the BRM database until all the events from the list appear in the database.

To prevent IMDB Cache DM from entering an infinite loop during the database polling process, you must specify a polling timeout value.

To specify a timeout value for batch polling operations, perform the following for each IMDB Cache instance:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.
2. Set the **dm_tt_unionsearch_poll_timeout** entry to the appropriate value:

```
-dm dm_tt_unionsearch_poll_timeout value
```

where *value* specifies the amount of time to wait, in milliseconds, before aborting the search procedure.

3. Save and close the file.
4. Restart the IMDB Cache DM instance.

Customizing External Applications for Real-Time Union Searches

If your BRM system uses an external application, you must configure it to perform union searches during real-time rating. To do so, customize your external application to pass the following input flist field in the call to the PCM_OP_SEARCH, PCM_OP_STEP_SEARCH, PCM_OP_GLOBAL_SEARCH, and PCM_OP_GLOBAL_STEP_SEARCH opcodes:

PIN_FLD_FLAGS field set to SRCH_UNION_TT. For example:

```
0 PIN_FLD_FLAGS INT[0] SRCH_UNION_TT
```

Customizing External Applications for Batch Polling Operations

If your BRM system uses an external application, you must configure it to perform batch polling operations. To do so, customize your external application to execute the **fm_utils_poll_tt** function as part of the **fm_utils.so** library.

FM_UTILS_POLL_TT

This function polls the BRM database to determine whether it has received all of the batch-rated events from Oracle IMDB Cache.

Syntax

```
Void
fm_utils_poll_tt(
    cm_nap_connection_t    *connp,
    int32                  flags,
    pin_flist_t             *s_flistp,
    pin_flist_t             **o_flistpp,
    pin_errbuf_t            *ebufp)
```

Parameters

- *Flags* indicates the polling type. No flag indicates multi-node polling, and the PCM_OPFLG_SEARCH_ONE_PARTITION flag indicates single-node polling. The default is multi-node polling.

- *s_flistp* is a pointer to the input flist. The input flist must use the following format:

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /poll -1 0
0 PIN_FLD_ARGS                ARRAY [1] allocated 20, used 2
1   PIN_FLD_ARG_TYPE          ENUM [0] 0
1   PIN_FLD_CLASS_NAME        STR [0] "/event"
```

- *o_flistp* is a pointer to the output flist. The output flist must use the following format:

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /poll -1 0
0 PIN_FLD_MOD_TIME            STR [0] "1298951268"
0 PIN_FLD_PROC_STATUS         INT [0] 0
```

Note: In the PIN_FLD_PROC_STATUS field, a value of **0** indicates that the data in the BRM database is synchronized with Oracle IMDB Cache. A value of **-1** indicates that the data in the BRM database is out-of-sync with Oracle IMDB Cache.

- *ebufp* is a pointer to the error buffer.

Customizing External Applications for Logical Partitions

If your Oracle IMDB Cache system includes logical partitions, you must customize any external applications to perform the following:

- Retrieve data from multiple logical partitions. See ["Retrieving Data Across Logical Partitions"](#).
- Search for data across multiple logical partitions. See ["Searching for Data Across Logical Partitions"](#).

For more information, see ["About Finding Data in Logical Partitions"](#).

Retrieving Data Across Logical Partitions

Any external application that retrieves data from your BRM system must specify when to retrieve data from only one logical partition and when to retrieve data from multiple logical partitions.

By default, the PCM_OP_READ_OBJ and PCM_OP_READ_FLDS opcodes automatically retrieve data from one logical partition only. To retrieve data from multiple logical partitions, you must customize your external application to pass the PCM_OPFLG_SEARCH_PARTITIONS flag in the opcode call.

For example, for C applications:

```
PCM_OP(ctxp, PCM_OP_READ_OBJ, PCM_OPFLG_SEARCH_PARTITIONS, input_flistp, &return_flistp, ebufp);
```

For example, for Java applications:

```
FList output = conn.opcode(PortalOp.READ, PortalContext.OPFLG_SEARCH_PARTITIONS, input);
```

Searching for Data Across Logical Partitions

Any external application that performs a search on your BRM system must specify when to search all logical partitions in a schema and when to search only one logical partition in a schema. To do so, you pass a flag in the call to the search opcodes.

The flag to send and the scenarios in which to pass the flag depends on whether the opcode call is made inside or outside of a transaction:

- **Inside of a transaction:** By default, IMDB Cache DM searches only the current logical partition when a search is executed inside of a transaction.

To customize your external application to request a search across all logical partitions in a call inside a transaction, pass the `PCM_OPFLG_SEARCH_PARTITIONS` flag in the call to the search opcode.

- For example, for C applications:

```
PCM_OP(ctxp, PCM_OP_SEARCH, PCM_OPFLG_SEARCH_PARTITIONS, input_flistp,
&return_flistp, ebufp);
```

- For example, for Java applications:

```
FList output = conn.opcode(PortalOp.SEARCH, PortalContext.OPFLG_SEARCH_
PARTITIONS, input);
```

- **Outside of a transaction:** By default, IMDB Cache DM automatically searches across all logical partitions when a search is executed outside of a transaction.

To customize your external application to request a search on only one logical partition in a call outside of a transaction, pass the `PCM_OPFLG_SEARCH_ONE_PARTITION` flag in the call to the search opcode.

- For example, for C applications:

```
PCM_OP(ctxp, PCM_OP_SEARCH, PCM_OPFLG_SEARCH_ONE_PARTITION, input_flistp,
&return_flistp, ebufp);
```

- For example, for Java applications:

```
FList output = conn.opcode(PortalOp.SEARCH, PortalContext.OPFLG_SEARCH_ONE_
PARTITION, input);
```

[Table 22–1](#) lists the BRM search opcodes that support global processing searches in IMDB Cache-enabled systems.

Table 22–1 *BRM Search Opcodes*

Opcode	Description
PCM_OP_SEARCH	Searches the database and returns all of the results simultaneously.

Table 22–1 (Cont.) BRM Search Opcodes

Opcode	Description
PCM_OP_STEP_SEARCH PCM_OP_STEP_NEXT PCM_OP_STEP_END	<p>Searches the database and returns the results in discrete chunks. Use this opcode when searching a single schema.</p> <p>You start a step search by calling the PCM_OP_STEP_SEARCH opcode, which initiates a step search and retrieves the first set of results. Then, you call the PCM_OP_STEP_NEXT opcode for each subsequent set of results. At the end of the search, you call the PCM_OP_STEP_END opcode to retrieve the last set of results.</p> <p>You pass the flag in the call to the PCM_OP_STEP_SEARCH opcode only.</p>
PCM_OP_GLOBAL_SEARCH	<p>Searches for data across multiple schemas and returns all of the results simultaneously.</p>
PCM_OP_GLOBAL_STEP_SEARCH PCM_OP_GLOBAL_STEP_NEXT PCM_OP_GLOBAL_STEP_END	<p>Searches for data across multiple schemas and returns the results in discrete chunks.</p> <p>You start a global step search by calling the PCM_OP_GLOBAL_STEP_SEARCH opcode, which initiates a step search and retrieves the first set of results. Then, you call the PCM_OP_GLOBAL_STEP_NEXT opcode for each subsequent set of results. At the end of the search, you call the PCM_OP_GLOBAL_STEP_END opcode to retrieve the last set of results.</p> <p>You pass the flag in the call to the PCM_OP_GLOBAL_STEP_SEARCH opcode only.</p>

For more information about:

- The BRM search opcodes, see "Searching for Objects in BRM Databases" in *BRM Developer's Guide*.
- Calling opcodes, see "Calling PCM Opcodes" in *BRM Developer's Guide*.

Configuring Your Event Tables for BRM Reports

To enable you to successfully run BRM reports on an IMDB Cache-enabled system, the IMDB Cache Manager installer automatically creates an event table, a temporary table, and a table view for each of the following objects:

- `/event/activity/content`
- `/event/activity/telco`
- `/event/session/dialup`
- `/event/session/telco`
- `/event/session/telco/gsm`
- `/event/session/telco/gprs`
- `/event/session/telco/gprs/master`
- `/event/session/telco/gprs/subsession`
- `/event/session/telco/imt`
- `/event/session/telco/pdc`

You must generate temporary tables and views for any other event tables in your system with residency type 301 or 302 by running the **create_temp_tables_proc()** stored procedure. For example, you run the stored procedure whenever you perform the following tasks:

- Create custom event tables with residency type 301 or 302.
- Install any BRM optional manager and want to run its associated report.

To configure your custom and optional manager event tables for BRM reports:

1. Log in to SQL*Plus as the **pin** user.
2. Run the following stored procedure:

```
call create_temp_tables_proc()
```

Configuring How to Store Reservations

You must configure BRM to store reservations in **/balance_group** objects, which are created in the subscriber cache group. To do so, set the **balance_coordinator** CM configuration file entry to **1**.

Because Oracle IMDB Cache does not support migration of local tables across grid members, **/reservation_list** cannot be used to track reservations for resource sharing groups. For this reason, it is recommended to set **balance_coordinator** to **1** in an IMDB Cache-enabled system so that reservations are tracked in **/balance_group** objects.

To configure BRM to store reservations in **/balance_group** objects:

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

```
- cm balance_coordinator 1
```
3. Save and close the file.

Setting the Stacksize Limit

On the system where IMDB Cache DM is installed, set the **stacksize** limit to **unlimited**.

To set the stacksize limit:

1. Log on to the system where IMDB Cache DM is installed.
2. Enter the following command:

```
limit stacksize unlimited
```
3. Verify the stacksize limit by entering the following command:

```
limit stacksize
```

Setting Database Schema Status for Oracle IMDB Cache Systems

Database schema status determines whether a database schema or logical partition is available for account creation. Databases can be set to open, closed, or unavailable. For more information, see ["How Accounts Are Assigned to the Logical Partition"](#).

To change the status of a database schema or logical partition, 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.

To set or change the database schema or logical partition status:

1. Go to the *BRM_home/apps/multi_db* directory and open the **config_dist.conf** file in a text editor.

2. Change the values in the STATUS entries:

```
DB_NO = "0.0.0.1" ;      # 1st database schema configuration block
PRIORITY = 1 ;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "OPEN" ;
DB_NO = "0.1.0.1" ;      # 1st database schema configuration block with logical
partition
PRIORITY = 1 ;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "OPEN" ;
DB_NO = "0.0.0.2" ;      # 2nd database schema configuration block
PRIORITY = 2;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN";
DB_NO = "0.1.0.2" ;      # 2nd database schema configuration block with logical
partition
PRIORITY = 1;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN";
```

Note: If your system contains multiple database schemas and logical partitions, create a new set of entries for each additional database schema and each logical partition.

3. Save and close the file.
4. Verify that the **pin_config_distribution** utility is not running.
5. Go to the *BRM_home/apps/multi_db* directory and run the **load_config_dist** utility.

Caution: 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.

6. Stop and restart all CMs.

Setting the Database Schema Priority for Oracle IMDB Cache Systems

Database schema priority determines when customer accounts are created on a particular database schema relative to other database schemas. BRM assigns accounts to an open database schema with the highest priority number.

If all database schemas have the same priority, BRM chooses an open database schema at random each time it assigns accounts. This distributes accounts evenly across all database schemas.

Oracle recommends assigning:

- Different priorities to each database schema.
- The same priority to all logical partitions within a database schema.

Note: If your system contains multiple database schemas and multiple logical partitions, create a new set of entries for each additional database schema and logical partition.

To set or change a database schema's priority:

1. Go to the *BRM_home/apps/multi_db* directory and open the **config_dist.conf** file in a text editor.
2. Edit the PRIORITY entries.

In the following example, BRM creates accounts on database schema 0.1.0.2 because it has the highest priority setting of all open database schemas.

```
DB_NO = "0.0.0.1" ;    # 1st database configuration block
PRIORITY = 1 ;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "OPEN" ;
DB_NO = "0.1.0.1" ;    # 1st database configuration block with logical partition
PRIORITY = 1;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "OPEN" ;
DB_NO = "0.0.0.2" ;    # 2nd database configuration block
PRIORITY = 2;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN" ;
DB_NO = "0.1.0.2" ;    # 2nd database configuration block with logical partition
PRIORITY = 1;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN";
DB_NO = "0.0.0.3" ;    # 3rd database configuration block
PRIORITY = 3;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "CLOSED" ;
DB_NO = "0.1.0.3" ;    # 3rd database configuration block with logical partition
PRIORITY = 1;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN"
```

3. Save and close the file.
4. Verify that the **pin_config_distribution** utility is not running.

Caution: The **load_config_dist** utility overwrites all distributions that are already in the database. If you are updating distributions or adding new ones, beware that you cannot load only the new and changed distributions.

5. Go to the *BRM_home/apps/multi_db* directory and run the **load_config_dist** utility.

Note: The **load_config_dist** utility requires a configuration file.

6. Stop and restart all CMs.

Generating the BRM Cache Group Schema

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) `pin_tt_schema_gen` utility, how to configure it, and how to use it.

About Generating the BRM Cache Group Schema

You generate the BRM cache group schema by using the `pin_tt_schema_gen` utility. This utility performs the following tasks:

- Generates the `tt_schema.sql` file used to create the BRM cache group schema
- Generates the `tt_load.sql` file used to load BRM objects into the cache groups
- Generates the `tt_drop.sql` file used to drop the BRM cache group schema
- Creates missing indexes and not null constraints on the BRM database

You first run the `tt_schema.sql` script on Oracle IMDB Cache to create the BRM cache groups schema. After running `tt_schema.sql`, you run the `tt_load.sql` script to preload the corresponding BRM objects from the BRM database into the cache groups.

Note:

- In a high-availability system with Oracle Clusterware, Oracle Clusterware automatically duplicates the complete data store from the active to the standby node.
 - In a high-availability system without Oracle Clusterware, you must run `tt_schema.sql` on both the active and the standby IMDB Cache nodes to create the schema for the BRM cache groups. You run `tt_load.sql` only on the active node, and the Oracle IMDB Cache replication agent replicates the data to the standby node.
-

To drop the BRM cache groups schema and the tables created in the cache groups, run the `tt_drop.sql` script. This will delete all BRM objects from the cache group. Data in the cache propagated to the BRM database before running `tt_drop.sql` persists in the BRM database.

Creating and Initializing Your Cache Group Schema

To create and initialize your schema:

1. [Configuring the `pin_tt_schema_gen.values` File.](#)
2. [Generating Your Schema and Load SQL Scripts.](#)

Configuring the `pin_tt_schema_gen.values` File

The `pin_tt_schema_gen` utility uses the existing BRM database schema to generate the `tt_schema.sql`, `tt_load.sql`, and `tt_drop.sql` scripts based on the specifications in the `pin_tt_schema_gen.values` file.

When you configure the `pin_tt_schema_gen.values` file, you specify values such as the following:

- The database name, user name, and password for the BRM database
- The database number to specify where to load the data
- The cache group definitions
- The cache group aging policy
- The transient tables

[Table 23–1](#) describes the entries in the `BRM_home/sys/pin_tt_schema_gen.values` file.

Table 23–1 `pin_tt_schema_gen.values` File Entries

Entry	Description
<code>\$MAIN_DB{'alias'}</code>	Specifies the database alias name for the BRM database defined in the <code>tnsnames.ora</code> file.
<code>\$MAIN_DB{'user'}</code>	Specifies the user name for the BRM database. For multischema systems, enter the user name for the database schema.
<code>\$MAIN_DB{'password'}</code>	Specifies the password for the BRM database user. For multischema systems, enter the password for the database schema user.
<code>\$fileName</code>	Specifies the name of the generated SQL file that can be used to create the schema definitions for the BRM cache groups. The default is <code>tt_schema.sql</code> . For multischema systems, append the database number for the schema to the file name. For example: <code>tt_schema_0.0.0.2.sql</code>
<code>\$droppingDatastoreSqlFile</code>	Specifies the name of the generated SQL file that can be used to drop the schema definitions for the BRM cache groups. The default is <code>tt_drop.sql</code> .
<code>\$loadingCachegroupSqlFile</code>	Specifies the name of the generated SQL file that can be used to load the data into the BRM cache groups. The default is <code>tt_load.sql</code> .
<code>\$logfile</code>	Specifies the name of the log file that contains the error logs for the <code>pin_tt_schema_gen</code> utility. The default is <code>pin_tt_schema_gen.log</code> .
<code>\$primary_db_user_name</code>	Specifies the name of the primary database user or the primary database schema user. This is required in a multischema system.
<code>\$logical_partitioning_enabled</code>	Specifies whether logical partitioning is enabled in Oracle IMDB Cache. The default is No .
<code>%cache_group_info</code>	Specifies cache group information. For details about this entry, see "Defining Cache Group Information" .

Table 23–1 (Cont.) *pin_tt_schema_gen.values* File Entries

Entry	Description
%cache_group_info_specific	Specifies information for special cases in which you must create the cache group definition for each table or class. For details about this entry, see "Defining Cache Group Information for Special Cases" .
%on_delete_cascade_def	Specifies that when rows containing referenced key values are deleted from a parent table, rows in child tables with dependent foreign keys are also deleted. For details about this entry, see "Setting On Delete Cascade" .
@lru_config_def	Specifies the least-recently-used (LRU) aging policy for the cache group. The default is ('.30', '.50', '1'). For details about this entry, see "Defining the Cache Group Aging Policy" .
@local_tables_class_def	Specifies transient tables. For details about this entry, see "Defining Transient Local Tables" .
\$use_timesten_datatypes	Specifies whether you want to change the NUM field type to TT_INTEGER and TT_BIGINT. The default is Yes . For details about this entry, see "Supporting Oracle IMDB Cache Data Types" .
%tt_load_cache_group_def	Specifies the cache groups in which to load data. For details about this entry, see "Generating the Load SQL Script" .
\$db_no_for_load_sql	Specifies the logical partition database number. This entry can be used if your Oracle IMDB Cache has multiple logical partitions. This enables you to generate separate SQL files that can be used to load the data for each logical partition or database number. Note: You must set logical_partitioning_enabled to Yes to use this entry. For details about this entry, see "Defining the Logical Partition Database Number" .
@specific_sql_query	Specifies additional SQL statements to be added in the generated schema definition SQL file.

The **pin_tt_schema_gen.values** file includes syntax information and specifications for the default BRM cache groups. Use the information in the following sections to supplement the information in **pin_tt_schem_gen.values**.

Defining Cache Group Information

You define each cache group by using the **%cache_group_info** entry. This entry uses the following syntax:

```
%cache_group_info = (cache_group_name => {
'CLASSES' => [class1, class2, ..... classN ],
'CG_PROPERTIES' => [cache_group_type, baseTableName, optionalAging, primary_key,
'USE_GLOBAL_CACHE_GROUP'],
'FK_DEF' => ['DEFAULT' => foreignKey, referredBaseTable, fieldOfReferredBaseTable',
```

```
'baseTableName_N
=> foreignKey, referredBaseTable, fieldOfReferredBaseTable'] } ,....);
```

Table 23–2 defines the parameters in the above syntax.

Table 23–2 Parameters for the %cache_group_info Entry

Parameter	Description
cache_group_name	Specifies the name of the cache group.
'CLASSES'	Specifies the BRM storable classes in the cache group, separated by commas. For example: <pre>'CLASSES' => ['/account', '/balance_group', '/bill', '/billinfo'],</pre>
'CG_PROPERTIES'	Specifies the properties of the cache group where: <ul style="list-style-type: none"> cache_group_type specifies the cache group type. baseTableName specifies the root table in the cache group. (Optional) optionalAging is used to specify whether aging is disabled for the cache group. Aging is enabled by default. primaryKey specifies the root table primary key. USE_GLOBAL_CACHE_GROUP is specified when logical partitioning is enabled. For example: <pre>'CG_PROPERTIES' => ['ASYNCHRONOUS WRITETHROUGH' # Cache Group Type 'ACCOUNT_T', # RootTable 'POID_ID0', # Primary Key 'USE_GLOBAL_CACHE_GROUP'],</pre>
'FK_DEF'	(Optional) Used to define the foreign key for the subsequent base table of the subsequent class. For example: <pre>'FK_DEF' => ['DEFAULT => ACCOUNT_OBJ_ID0,ACCOUNT_T,POID_ID0', # Default for other root table of subsequent classes. 'JOURNAL_T => ITEM_OBJ_ID0,ITEM_T,POID_ID0]</pre>

Defining Cache Group Information for Special Cases

You define cache group information for each BRM database table by using the %cache_group_info_specific entry. This entry uses the following syntax:

```
%cache_group_info_specific = ( table_name => [ cache_group_name, primary_key,
cache_group_type, 'AGING_OFF', 'USE_PRIMARY_DB', 'USE_GLOBAL_CACHE_GROUP' ],
..... );
```

Table 23–3 describes the parameters in the above syntax.

Table 23–3 Parameters for the %cache_group_info Entry

Parameter	Description
table_name	Specifies the name of the cache table.

Table 23–3 (Cont.) Parameters for the %cache_group_info Entry

Parameter	Description
<i>cache_group_name</i>	Specifies the name of the cache group.
<i>primary_key</i>	The primary key for the specified table.
<i>cache_group_type</i>	Specifies the cache group type.

Setting On Delete Cascade

You specify the table on which to apply ON DELETE CASCADE by using the %on_delete_cascade_def entry. This entry uses the following syntax:

```
%on_delete_cascade_def = ( 'cache_group_name' =>
['table1','table2',...,'tableN'],.....);
```

where:

- *cache_group_name* specifies the name of the cache group schema.
- *tableN* is the table name.

For example:

```
%on_delete_cascade_def = ( 'ACCOUNT_CG' => ['JOURNAL_T'] );
```

Defining the Cache Group Aging Policy

By default, IMDB Cache Manager checks the memory space usage of the Oracle IMDB Cache shared-segment at one-minute intervals. When the memory usage exceeds 50%, IMDB Cache Manager purges the least-recently-used (LRU) objects from the cache groups until the memory space usage is below 30%.

Note: Aging is enabled by default for Dynamic AWT cache groups. To disable aging for any cache group, you must specify AGING_OFF in the cache group definition. See ["Defining Cache Group Information"](#) for more information.

You define the Oracle IMDB Cache aging policy by using the @lru_config_def entry. This entry uses the following syntax:

```
@lru_config_def = ('LowUsageThreshold', 'HighUsageThreshold', 'AgingCycle' );
```

where:

- *LowUsageThreshold* is the memory space usage in Oracle IMDB Cache below which the aging policy is deactivated. The default is **.30**.
- *HighUsageThreshold* is the memory space usage in Oracle IMDB Cache above which the aging policy is activated. The default is **.50**.
- *AgingCycle* is the frequency (in minutes) that the memory space usage is checked. The default is **1**.

For detailed information about implementing aging on a cache group, see *Oracle In-Memory Database Cache User's Guide*.

Defining Transient Local Tables

Transient tables are created as local tables in Oracle IMDB Cache. You define which storable classes to create as local tables by using the `@local_tables_class_def` entry. This entry uses the following syntax:

```
@local_tables_def = ('class1', 'class2', ..... , 'classN');
```

where *classN* is a BRM storable class.

The default entry is set to the following:

```
@local_tables_class_def = ('/reservation','/reservation/active','/reservation_
list','/active_session','/active_session/telco','/active_
session/telco/gprs','/active_session/telco/gprs/master','/active_
session/telco/gprs/subsession','/active_session/telco/gsm');
```

Supporting Oracle IMDB Cache Data Types

You can reduce the amount of data stored in Oracle IMDB Cache by converting all BRM fields with `NUMBER(38)` data type to a `TT_INTEGER` or `TT_BIGINT` data type. The Oracle IMDB Cache integer data types `TT_INTEGER` and `TT_BIGINT` require less memory than the BRM integer data type `NUMBER(38)`.

- `NUMBER(38)` requires 22 bytes of storage.
- `TT_INTEGER` requires 4 bytes of storage.
- `TT_BIGINT` requires 8 bytes of storage.

To convert all fields with a `NUMBER(38)` data type to a `TT_INTEGER` or `TT_BIGINT` data type, set the `$use_timesten_datatypes` entry to YES:

```
$use_timesten_datatypes = "YES";
```

When this entry is set to **NO** or missing, fields with a `NUMBER(38)` data type are left unchanged.

Generating the Load SQL Script

You define the load criteria for generating the `tt_load.sql` script by using the `%tt_load_cache_group_def` entry. This entry uses the following syntax:

```
%tt_load_cache_group_def = ('Cache_group_name', => 'no_of_rows_to_commit,
condition, no_of_threads')
```

```
%tt_load_cache_group_def = ('Cache_group_name, LOCAL'=> 'no_of_rows_to_commit,
condition, no_of_threads')
```

where:

- *Cache_group_name* is the name of the cache group.
- *no_of_rows_to_commit* is the number of rows to insert into the cache group before committing the work. It must be a non-negative integer. If *no_of_rows_to_commit* is 0, the entire statement is executed as one transaction.
- (Optional) *condition* is the search condition used to retrieve the target rows from the BRM database.
- (Optional) *no_of_threads* is the number of loading threads that run concurrently for parallel loading. One thread performs the bulk fetch from the BRM database and the remaining threads perform the inserts into Oracle IMDB Cache. Each thread uses its own connection or transaction.

The minimum value for *no_of_threads* is **2**. The maximum value is **10**. If you specify a value greater than 10, Oracle IMDB Cache database assumes the value is **10**.

For a high-availability system, you must configure **pin_tt_schema_gen.values** to generate **tt_load.sql** for the specific Oracle IMDB Cache database node and then run **tt_load.sql** against the respective node.

Defining the Logical Partition Database Number

You specify the logical partition database number on which to run the **tt_load.sql** script by using the **\$dm_no_for_load_sql** entry. This entry uses the following syntax:

```
$db_no_for_load_sql = "db_number";
```

where *db_number* is the logical partition database number.

For example, suppose there are two database schemas with two logical partitions for each schema as follows:

- Schema 1 has logical partitions 0.0.0.1 and 0.1.0.1
- Schema 2 has logical partitions 0.0.0.2 and 0.1.0.2

On the machine with logical partition 0.0.0.1, set **\$db_no_for_load_sql** to "0.0.0.1". Then generate **tt_load.sql** using **pin_tt_schema_gen -l**.

On the machine with logical partition 0.1.0.1, set **db_no_for_load_sql** to "0.1.0.1".

On the machine with logical partition 0.0.0.2, set **\$db_no_for_load_sql** to "0.0.0.2".

On the machine with logical partition 0.1.0.2, set **\$db_no_for_load_sql** to "0.1.0.2".

Note: For logical partitioning support, set **logical_partitioning_enabled** to **YES**.

In a high-availability system with Oracle Clusterware, Oracle Clusterware automatically duplicates the complete data store from the active node to the standby node.

In a high-availability system without Oracle Clusterware, you must

- Run **tt_schema.sql** on both the active and the standby IMDB Cache nodes to create the schema for the BRM cache groups.
- Run **tt_load.sql** only on the active node. The Oracle IMDB Cache replication agent replicates the data to the standby node.

See ["Using the Oracle IMDB Cache Grid to Partition Data"](#) for more information about logical partitioning.

Generating Your Schema and Load SQL Scripts

You can generate schema and load SQL scripts for a basic BRM system or a BRM system with logical partitioning.

Important: If you have existing BRM data created on a BRM system before IMDB Cache Manager was installed, you must run the **load_pin_uniqueness** utility before loading BRM objects into the Oracle IMDB Cache.

Generating Scripts for a Basic BRM System

To generate the schema and load SQL scripts:

1. If you have not already done so, modify the **pin_tt_schema_gen.values** file. See ["Configuring the pin_tt_schema_gen.values File"](#).

2. Run the following command:

```
source BRM_home/source.me.csh
```

3. Run the **pin_tt_schema_gen** utility with the **-a** parameter:

```
./pin_tt_schema_gen -a
```

Note: If you do not specify the values for **MAIN_DB{'user'}** and **MAIN_DB{'password'}** in the **pin_tt_schema_gen.values** file, the **pin_tt_schema_gen** utility prompts you to enter these values.

This updates the BRM database with unique indexes and non-null constraints and generates the following files:

- **tt_schema.sql**
- **tt_load.sql**
- **tt_drop.sql**

Generating Scripts for a BRM System with Logical Partitioning

To generate your scripts for logical partitions, perform the following steps for each database schema on your system:

1. Open the **pin_tt_schema_gen.values** file in a text editor.
2. Set the **\$logical_partitioning_enabled** entry to **Yes**.
3. Set the **\$db_no_for_load_sql** entry to the appropriate value for the *first* logical partition in your database schema.
4. Save and close the file.
5. Run the following command:

```
source BRM_home/source.me.csh
```

6. Run the **pin_tt_schema_gen** utility with the **-a** parameter:

```
./pin_tt_schema_gen -a
```

This generates your **tt_schema.sql**, **tt_drop.sql**, and **tt_load_0.M.0.N.sql** scripts, where *M* is the partition number and *N* is the database schema number.

7. Open the **pin_tt_schema_gen.values** file in a text editor.
8. Set the **\$db_no_for_load_sql** entry to the appropriate value for the *second* logical partition in your database schema.
9. Save and close the file.
10. Run the **pin_tt_schema_gen** utility with the **-l** parameter:

```
./pin_tt_schema_gen -l
```


This generates the **tt_load_0.M.0.N.sql** script for loading BRM data into your second logical partition.

Customizing IMDB Cache Manager

This chapter provides information for customizing Oracle Communications Billing and Revenue Management (BRM) for In-Memory Database (IMDB) Cache Data Manager (DM).

About Customizing the Cache Groups

The following sections describe how to customize the default BRM cache groups and how to add custom cache groups to the BRM cache groups schema.

When you modify an existing cache group definition, you must unload any data in the cache group, drop the existing cache group, create the cache group with a new definition, and reload the data. Therefore, this operation requires service downtime. See *Oracle In-Memory Database Cache User's Guide* for more information about cache group operations.

Customizing the Default BRM Cache Groups

You can modify the default BRM cache groups by adding custom storable objects or by modifying existing storable objects.

To customize the default BRM cache groups:

1. Create the custom storable class or modify the existing storable class. See ["Creating Custom Fields and Storable Classes"](#).
2. Edit the `pin_tt_schema_gen.values` file (`BRM_home/bin/pin_tt_schema_gen.values`) and modify the existing cache group definitions as needed. See ["Configuring the pin_tt_schema_gen.values File"](#).

No: All custom storable classes for subscriber data must be added to the existing subscriber cache group. See ["About Extending Tables in the Default BRM Cache Groups"](#) for more information.

3. Run the `pin_tt_schema_gen` utility with the `-t` parameter. This creates a schema SQL script that includes your custom tables and fields.
4. Using the TimesTen `ttIsql`, update the BRM cache groups schema in Oracle IMDB Cache as follows:
 - a. Stop the Oracle IMDB Cache replication agent.
 - b. Stop the cache agent.
 - c. Drop the cache group schema.

- d. Run the **tt_schema.sql** file to generate the schema with the new definitions.
 - e. Start the cache agent.
 - f. Start the replication agent.
5. Reload the cache group, if needed.
6. Restart the IMDB Cache DM instance.

About Extending Tables in the Default BRM Cache Groups

All custom storable classes for subscriber data must be added to the existing subscriber cache group definition to support resource sharing relationships. For example, if you define a custom service storable class such as **/service/voip**, the corresponding custom tables should be added in the subscriber cache group schema. Custom storable classes for other data types should be added to custom cache group definitions. See ["Creating a Custom Cache Group"](#).

All custom storable classes for transient objects should be added to the Oracle IMDB Cache local tables.

Creating a Custom Cache Group

You create custom cache groups manually by defining the custom cache group information in the **pin_tt_schema_gen.values** file, generating the schema SQL file, and then creating the schema in Oracle IMDB Cache. You can create the custom cache group using existing BRM storable objects or custom storable objects.

To create custom cache groups in Oracle IMDB Cache:

1. Create the custom storable class. See ["Creating Custom Fields and Storable Classes"](#).
2. Edit the **pin_tt_schema_gen.values** file (*BRM_home/bin/pin_tt_schema_gen.values*) and add a new cache group definition. See ["Generating the BRM Cache Group Schema"](#).
3. Run the **pin_tt_schema_gen** utility with the **-t** parameter to generate the **tt_schema.sql** script.
4. Using the TimesTen **ttIsql** tool, update the BRM cache groups schema in Oracle IMDB Cache as follows:
 - a. Stop the Oracle IMDB Cache replication agent.
 - b. Stop the cache agent.
 - c. Drop the cache group schema.
 - d. Run the **tt_schema.sql** file to generate the schema for the custom cache groups.
 - e. Start the cache agent.
 - f. Start the replication agent.
5. Load the cache group, if needed.
6. Restart the IMDB Cache DM instance.

See the *Oracle In-Memory Database Cache User's Guide* for more information on the different types of cache groups and how to define them.

Creating Custom Fields and Storable Classes

Use Developer Center to create custom fields and storable classes and to modify existing fields and storable classes in the BRM database.

For specific instructions on how to create or modify fields and storable classes, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

After you create custom field and classes, you must make them available to applications. For specific instructions, see "Making Custom Fields Available to Your Applications" in *BRM Developer's Guide*.

Assigning Custom Objects a Residency Value

When you define a custom storable class, you must also set the residency type for the class. The residency type specifies where the storable object resides in the BRM system. See the discussion on about the residency type in *BRM Concepts*.

Object residency types are predefined in the BRM data dictionary. To determine the residency type value for your custom storable class, you can run a SQL query on the BRM database to get the residency type values. For example, to get all the objects with residency type values 1, 5, or 7, you can run the query:

```
Select name, residency_type from dd_objects_t where residency_type in (1,5,7);
```

To set the residency type value for the custom storable class, you can run a SQL query on the BRM database to set the value. For example, to set the value for a custom service class, you can run the query:

```
update dd_objects_t set residency_type=1 where name like '/service%' and  
residency_type !=5 and residency_type !=7;
```

After setting the residency type value for the custom storable class, restart the IMDB Cache DM instance, and the Connection Manager (CM).

About Extending Event Type Storable Classes

When you define a custom event storable class, it inherits the attributes of the parent event storable class. Event attributes can be:

- Oracle IMDB Cache resident expanded object
- BRM resident expanded object

When you extend the base event class (*/event*), you define the subclass by setting the residency type.

Logical Partitioning and POID DB Translation

To retrieve only the database schema number from POID DB (excluding the logical partition number), use the PIN_GET_SCHEMA_NO macro.

PIN_GET_SCHEMA_NO

This macro returns the database schema number from the database number provided in the parameter. A database number is represented in the format *0.logical_partition_number.0.database_schema_number*. The PIN_GET_SCHEMA_NO macro extracts *database_schema_number* from the database number.

Syntax

PIN_GET_SCHEMA_NO (*Database_number*);

Parameters

Database_number

A 64-bit integer that represents a database number.

Return Values

Returns the database schema number.

Error Handling

This macro does not return any error.

Migrating Data to an Oracle IMDB Cache-Enabled BRM System

This chapter describes how to migrate data from an Oracle Communications Billing and Revenue Management (BRM) system to an Oracle In-Memory Database (IMDB) Cache-enabled BRM system.

Note: The information applies to customers who have created accounts on a BRM system *before* IMDB Cache Manager was installed; it does not apply to customers performing a new BRM installation whose original system setup includes IMDB Cache Manager.

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

- BRM architecture.
- IMDB Cache Data Manager.
- Oracle IMDB Cache.

For more information, see "BRM System Architecture" in *BRM Concepts*.

About Migrating Data to an Oracle IMDB Cache-Enabled BRM System

If you have an existing BRM system, to use BRM with Oracle IMDB Cache, you must install IMDB Cache Manager and migrate your data on a BRM system to an Oracle IMDB Cache-enabled BRM system.

Migrating your BRM system to an Oracle IMDB Cache-enabled system includes:

- Preparing existing account data (*created* in your BRM system before using an Oracle IMDB Cache environment) for an Oracle IMDB Cache-enabled BRM environment.
- Migrating accounts from your BRM database to Oracle IMDB Cache logical partitions.
- Migrating a BRM TIMOS environment to the Oracle IMDB Cache environment

For an overview of the tasks required to migrate your BRM system to an Oracle IMDB Cache-enabled system, see "[Overview of the Migration Process](#)".

For information on how the migration process works, see the following:

- [How IMDB Cache Manager Migrates Your Existing BRM Accounts](#)

- [How IMDB Cache Manager Handles Subscriber Distribution for Account Migration](#)
- [How IMDB Cache Manager Handles POID Fields](#)

For information on how core BRM is modified after you install IMDB Cache Manager, see "[Changes to BRM after Installing IMDB Cache Manager](#)".

BRM Processes That Cannot Coexist with IMDB Cache Manager

When you migrate to an IMDB Cache-enabled system, the following BRM processes cannot coexist with IMDB Cache Manager:

- Oracle DM

IMDB Cache DM replaces the Oracle DM.

Important: During BRM 7.5 installation, the CM and Oracle DM are required and started up automatically by the **pin_setup** process.

When you configure a BRM system with IMDB Cache Manager, configure the CM to connect *only* to IMDB Cache DM.

- TIMOS DM

The functionality of IMDB Cache Manager and the Oracle IMDB Cache replaces the functionality of the TIMOS DM and the TIMOS DM reference object cache (ROC). If you have a BRM system with TIMOS DM, you must stop the TIMOS DM, CM, and Oracle DM processes before you perform certain phases of the upgrade.

How IMDB Cache Manager Migrates Your Existing BRM Accounts

This section explains how BRM and IMDB Cache Manager migrate the data in your BRM system to a BRM system with IMDB Cache Manager.

The Portal object ID (POID) of fields in a BRM system is in the format **0.0.0.Schema_No.** The POID of fields in a BRM system with IMDB Cache Manager is in the format **0.LogicalPartition_No.0.Schema_No.**

When the accounts in your BRM system migrate to a BRM system with IMDB Cache Manager, the upgrade process updates the POID to include the Oracle IMDB Cache logical partition number of the logical partition into which the accounts are migrated.

The **load_pin_uniqueness** utility:

- Sets and updates the logical partition number in the **/uniqueness** object POID that is stored in the BRM database so that it can be used later by the **tt_load.sql** script to load the data into IMDB Cache.
- Populates the **/uniqueness** table with uniqueness data for the accounts in your BRM system.
- Finds out which logical partition the accounts fit into and then updates the account POID database numbers in the **/uniqueness** table.

After **load_pin_uniqueness** figures out how to distribute the accounts among the logical partitions, the **tt_load.sql** script uses the data in the **/uniqueness** table to determine the Oracle IMDB Cache data stores in which to load the subscriber data for the accounts and loads the subscriber data.

Note: You run the **load_pin_uniqueness** utility only if you must migrate accounts that were created on a BRM system that was not IMDB Cache-enabled. After your accounts are migrated, you do not need to run this utility again.

How IMDB Cache Manager Handles Subscriber Distribution for Account Migration

Before accounts from your BRM system can be distributed among the logical partitions of Oracle IMDB Cache, they must be assigned to logical partitions.

The **load_pin_uniqueness** utility assigns the accounts from your BRM system to the logical partitions as follows:

- Finds accounts that are not updated in the **/uniqueness** table.
- Obtains the maximum account size and the current account size of the logical partition that the accounts may go into.
- Considers the maximum account size and the current account size of each logical partition and attempts to distribute accounts equally among logical partitions.
- If the logical partition reaches the limit, the remaining accounts are not distributed.
- Checks whether all the logical partitions of a schema can accommodate all the accounts of that schema.
- If all of the logical partitions of a schema cannot accommodate all of the accounts of that schema, throws an error message.
- Distributes accounts of a schema across all the logical partitions of the schema.

How IMDB Cache Manager Handles POID Fields

The BRM database and Oracle IMDB Cache do not store the logical partition number in the POID fields of their data. IMDB Cache Manager stores the logical partition number in its run-time memory. For the CM and its associated Facilities Modules (FMs) to know in which logical partition an account resides, the IMDB Cache DM does the following for all POID fields:

- Clears the logical partition number from the POID
- Writes the POID database numbers to the storage media
- Gets the POID database numbers from the storage media
- Updates the POID database numbers with the logical partition number in its memory
- Sends the resulting POID database numbers, which now include the logical partition number (of the current residing logical partition), to the CM

The IMDB Cache DM updates the POID fields with the logical partition number for all fields that store POIDs, including fields that store POIDs in string format.

Using the Same Logical Partition for Hierarchical Accounts

During billing, BRM migrates a parent account to the logical partition where the child account resides. After billing, BRM migrates the parent account and its related instances back to its original logical partition to avoid having a load imbalance among logical partitions. BRM uses the **/uniqueness** object for account lookup to obtain the

logical partition number of the parent account and move it back to that logical partition.

Frequent migration of accounts between logical partitions can degrade performance. For improved performance, Oracle recommends you load all hierarchical accounts into one logical partition.

Overview of the Migration Process

Important: If you are migrating from a legacy billing system, you must first migrate your data from the legacy system to BRM 7.5 before you begin the migration process to Oracle IMDB Cache.

If you have an existing BRM system, you perform a functional and technical upgrade to migrate to a BRM system that is Oracle IMDB Cache-enabled as follows:

1. Install BRM 7.5.
2. Stop the Connection Manager (CM) and Oracle Data Manager (DM) processes.
3. If you are migrating from a BRM TIMOS environment, stop the TIMOS DM.
4. Set up your Oracle IMDB Cache environment.
5. Install IMDB Cache Manager.
6. Start the IMDB Cache DM and CM processes.
7. Truncate or clean up the **/uniqueness** objects.

Note: This is applicable in a multischema environment only.

8. Restart the IMDB Cache DM process.
9. Stop the **config_pin_distribution** process.
10. Run the **load_pin_uniqueness** utility.
11. Configure the IMDB Cache DM, CM, and **load_pin_uniqueness** utility for migration. See ["Configuration Entries Required for Migration"](#).
12. Prepare accounts for distribution. See ["Preparing Accounts for Distribution into IMDB Cache Logical Partitions"](#).
13. Migrate from a TIMOS environment to an IMDB Cache environment. See ["Migrating a BRM TIMOS Environment to an IMDB Cache Environment"](#).
14. Load subscriber data into the IMDB Cache data stores. See ["Loading Subscriber Data into the Oracle IMDB Cache Data Stores"](#).

Configuration Entries Required for Migration

To configure your IMDB Cache DM, CM, and **load_pin_uniqueness** utility for migrating BRM data:

1. Open the IMDB Cache DM configuration file (*BRM_home/sys/dm_tt/pin.conf*) in a text editor.
2. Add the following entry:

```
- dm poids_in_string_format_list
PIN_FLD_ACCOUNT_NO,PIN_FLD_ITEM_POID_LIST,PIN_FLD_EVENT_POID_LIST,PIN_FLD_NEXT_
ITEM_POID_LIST,PIN_FLD_GL_REPORT_POID_LIST
```

This entry updates the POID fields with the Oracle IMDB Cache logical partition number.

3. Save and close the file.
4. Open the CM configuration file (*BRM_home/sys/cm/pin.conf*) in a text editor.
5. Add the following **uniqueness_login** entry:


```
- cm uniqueness_login 1
```
6. Add the following **use_legacy_uniqueness_population** entry:


```
- fm_cust use_legacy_uniqueness_population 0
```
7. Add the following **primary_db** entry:


```
- cm primary_db 0.0.0.1 / 0
```
8. Save and close the file.
9. Open the **load_pin_uniqueness** utility configuration file (*BRM_home/apps/multi_db/pin.conf*) in a text editor.
10. Add the following **is_timesten** entry:


```
- load_pin_uniqueness is_timesten 1
```
11. Add the following **per_schema_node_info** entry:


```
- load_pin_uniqueness per_schema_node_info DB_NO:DB_NO
```

where:

DB_NO:DB_NO is the per-schema logical partition information.

For example:

If the schema **0.0.0.1** has three logical partitions (**0.0.0.1**, **0.1.0.1**, and **0.2.0.1**), the entry is:

```
- load_pin_uniqueness per_schema_node_info 0.0.0.1:0.1.0.1:0.2.0.1
```
12. Add the following entry:


```
- load_pin_uniqueness -g
```

The **-g** parameter assigns all the group accounts to the same node. If you do not specify this parameter, group accounts are considered as nongroup accounts and the distribution follows the round-robin method.
13. Save and close the file.

Preparing Accounts for Distribution into IMDB Cache Logical Partitions

To prepare your existing account data for migration and distribution into available Oracle IMDB Cache logical partitions:

1. Run the **load_pin_uniqueness** utility.

You must run **load_pin_uniqueness** as described in ["Installing IMDB Cache Manager"](#).

2. Stop and restart the CM.
3. Verify that the **/uniqueness** object was loaded by using the Object Browser or by using the **robj** command with the **testnap** utility.

Your data is ready to be loaded into the Oracle IMDB Cache data stores. See ["Loading Subscriber Data into the Oracle IMDB Cache Data Stores"](#).

Migrating a BRM TIMOS Environment to an IMDB Cache Environment

This section describes configuration procedures you must follow if you use a BRM system with TIMOS DM.

To migrate a BRM TIMOS environment to an Oracle IMDB Cache environment:

1. Stop the CM.
2. Stop TIMOS DM.
3. Stop the Oracle DM.
4. Open the *BRM_home/sys/cm/pin.conf* file in a text editor.
5. Set the **dm_pointer** entry to point to the IMDB Cache DM host name and port number.

```
- cm dm_pointer 0.0.0.1 ip DMTT_Host Port
```

where *DMTT_Host* is the IMDB Cache DM host name.

Note: You must remove the TIMOS entry and add the IMDB Cache DM host name and port number.

6. Start the IMDB Cache DM.
7. Start the CM.
8. Run the **load_pin_uniqueness** utility.

Note: The transient objects are not migrated.

Loading Subscriber Data into the Oracle IMDB Cache Data Stores

To load subscriber data from the BRM database into the Oracle IMDB Cache data stores:

1. Run **pin_tt_schema_gen**, which generates the schema and load SQL files.
2. Connect to Oracle IMDB Cache using **ttIsql**.
3. Run the **tt_load.sql** script against the appropriate Oracle IMDB Cache logical partition.

For detailed instructions, see ["Generating the BRM Cache Group Schema"](#).

Changes to BRM after Installing IMDB Cache Manager

This section describes changes made to the base BRM 7.5 release after you install IMDB Cache Manager.

Opcode Changes

After you install IMDB Cache Manager (the **if_tt_enabled** flag is enabled), the opcode modifications listed in [Table 25–1](#) occur:

Table 25–1 Opcode Modifications for IMDB Cache Manager

Changed Opcode	Changed Opcode Flow in an Oracle IMDB Cache Environment
PCM_OP_ACT_FIND	Uses the /uniqueness object to obtain account and service data for a given account login. Performs uniqueness lookup in a single schema environment. Does not perform global search in a single schema environment.
PCM_OP_CUST_COMMIT_CUSTOMER	Updates the /uniqueness table.
PCM_OP_CUST_MODIFY_CUSTOMER	Updates the /uniqueness table.
PCM_OP_CUST_SET_LOGIN	Updates the /uniqueness table.
PCM_OP_CUST_MODIFY_SERVICE	Updates the /uniqueness table.
PCM_OP_CUST_DELETE_ACCT	Updates the /uniqueness table.
PCM_OP_TCF_AAA_AUTHORIZE	Uses the /uniqueness table for service lookup in a single schema environment.
PCM_OP_TCF_AAA_REAUTHORIZE	Uses the /uniqueness table for service lookup in a single schema environment.
PCM_OP_TCF_AAA_STOP_ACCOUNTING	Uses the /uniqueness table for service lookup in a single schema environment.

Utility Changes

After you install IMDB Cache Manager (the **if_tt_enabled** flag is enabled), the utility modifications listed in [Table 25–2](#) occur:

Table 25–2 Utility Modifications for IMDB Cache Manager

Utility	Change
load_pin_uniqueness	Distributes accounts among Oracle IMDB Cache logical partitions.
pin_tt_schema_gen	Generates the load command (tt_load.sql) based on uniqueness lookup information (the /uniqueness table entry).

Part VI

Partitioning and Managing BRM Tables

Part VI describes how to partition the Oracle Communications Billing and Revenue Management (BRM) tables, how to purge data, and how to generate virtual columns on event tables. It contains the following chapters:

- [Partitioning Tables](#)
- [Converting Nonpartitioned Classes to Partitioned Classes](#)
- [About Purging Data](#)
- [Generating Virtual Columns on Event Tables](#)

Partitioning Tables

This chapter explains how to organize your Oracle Communications Billing and Revenue Management (BRM) database by using partitioned tables.

Before partitioning any tables, you should be familiar with the following:

- Basic BRM concepts. See "Introducing 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 "Enabling Different Classes for Partitioning during Installation" in *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, see the following:

- When installing BRM, see "Enabling Different Classes for Partitioning during Installation" in *BRM Installation Guide*.
- After installing BRM, see ["Converting Additional 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"](#).

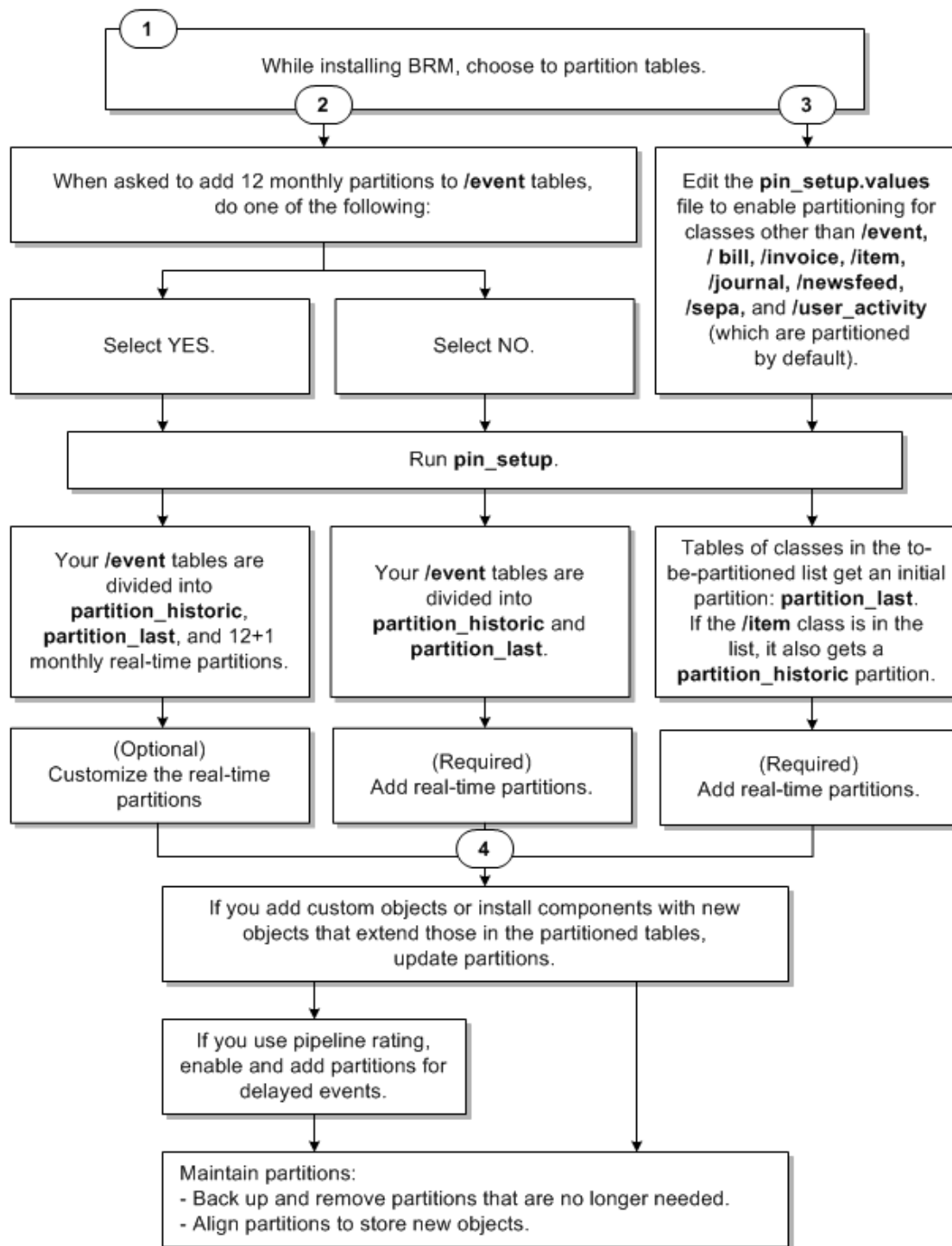
Note: Rule-based optimization is not available for partitioned tables and indexes. See "Using Rule-Based Optimization versus Cost-Based Optimization" in *BRM Installation Guide*.

Overview of Enabling Partitioning during Installation

[Figure 26-1](#) provides an overview of how to set up and use partitioning when you install BRM. All steps before you customize or add partitions must be done during installation.

For information about enabling partitioning after installation, see ["About Converting Nonpartitioned Classes to Partitioned Classes"](#).

Figure 26–1 Overview of Enabling Partitioning during Installation



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.

Important: 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.

If you use pipeline batch rating, Rated Event (RE) Loader loads delayed events into the BRM database. You can enable partitioning for delayed events and create partitions for the delayed events. 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 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. 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.

Caution: 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 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, you use the **partition_utils** utility to enable delayed-event partitions for specified event storable classes. See ["Syntax for 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. See ["Syntax for Adding 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

The following tables show the possible partitioning schemes.

If you do not enable partitions when installing BRM, every object table has no partitions as shown in [Table 26-1](#).

Table 26-1 No Partitions

Table
Object table

An object table designated for partitioning is partitioned as shown in [Table 26-2](#) if you enable partitions but do not create the 12 default partitions.

Table 26-2 Real-Time Partitioning Enabled, No Default Partitions

Partition	Partition
partition_historic (event and item tables only)	partition_last

Caution: Do not use the partitioning scheme shown in [Table 26-2](#) in a production system because all nonhistoric 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 26-3](#) if you enable partitions and accept the 12 default partitions.

Table 26-3 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 26–4](#) if you enable partitions and create custom partitions.

Table 26–4 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 26–5](#) if you enable partitions without the 12 default partitions and then enable delayed-event partitions.

Table 26–5 Real-Time Partitioning Enabled, Delayed Partitioning Enabled, No Default Partitions

Partition	Partition	Partition
partition_historic	partition_last_pin	partition_last

Caution: Do not use the partitioning scheme shown in [Table 26–5](#) in a production system because all nonhistoric 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 26–6](#) if you enable partitions, create real-time partitions, and enable delayed-event partitions.

Table 26–6 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

Caution: Do not use the partitioning scheme shown in [Table 26–6](#) in a production system because all nonhistoric delayed events are stored in **partition_last**. You must add delayed partitions.

An event table is partitioned as shown in [Table 26–7](#) if you enable partitions, do not create real-time partitions, and enable and create delayed partitions.

Table 26–7 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

Caution: Do not use the partitioning scheme shown in [Table 26–7](#) in a production system because all nonhistoric real-time events are stored in **partition_last_pin**. You must add real-time partitions.

An event table is partitioned as shown in [Table 26–8](#) if you enable partitions, create real-time partitions, and enable and create delayed partitions.

Table 26–8 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	DP_1	DP_2	DP_3 through DP_(N-1)	DP_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

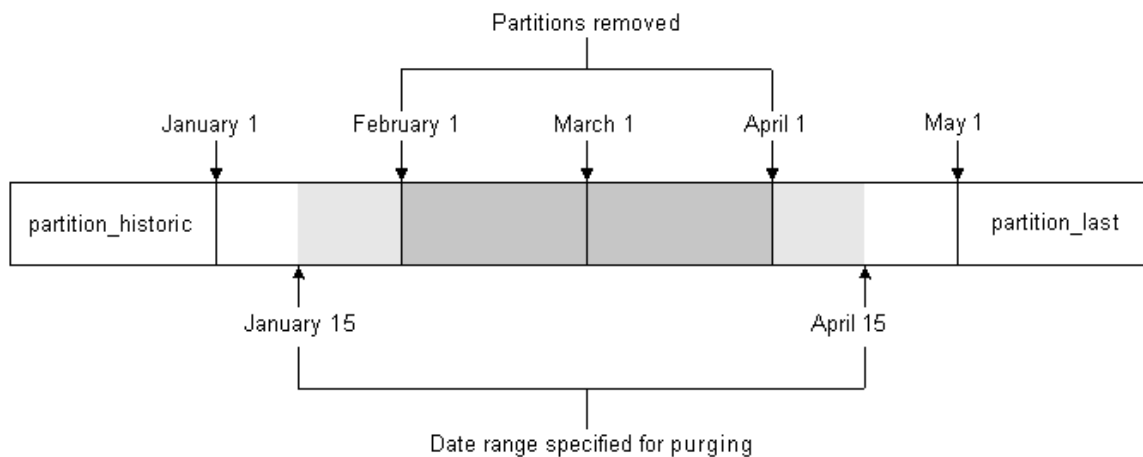
When you purge objects by removing their partitions, you specify a start date and an end date to identify the objects to be removed. For example, you can purge objects created from January 15 to March 15.

Important: 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"](#).

Figure 26–2 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 26–2 Purging of Complete Partitions



For more information, see ["Purging Objects by Removing Partitions"](#).

Important: 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"](#).

Important: Purging partitions is considered safer than removing them.

About Running the partition_utils Utility

Most partition management tasks are accomplished by using the **partition_utils** utility.

Caution: After using your most current version of the **partition_utils** utility, *do not* use the 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.

Caution: 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, or enable partitions, you can use the **-p** parameter to write an 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.

Caution: 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.myserver.com";  
$MAIN_DB{'user'} = "pin";  
$MAIN_DB{'password'} = "pin";
```

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 unless the **enable** operation is executed.

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.

Important:

- 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 by stopping Pipeline Manager and RE Loader.

Important: 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.

Note: 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, 2015, and one partition for events up to July 27, 2015. Each partition includes events for a two-week period:

```
partition_utils -o add -t realtime -s 07272015 -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 07272015 -u week -w 2 -q 5 -f
```

The following command adds six daily event partitions starting on July 27, 2015, and one partition for events up to July 27, 2015. Because the **-w** parameter is not used, each parameter is one day long:

```
partition_utils -o add -t delayed -s 07272015 -u day -q 6
```

The following command adds 12 monthly real-time partitions for the journal storable class starting on July 27, 2015, and one partition for journal up to July 27, 2015:


```
partition_utils -c /journal -o add -t realtime -s 07272015 -u month -q 12
```

The following command creates five weekly partitions for the journal storable class starting on July 27, 2015, and one partition for journal up to July 27, 2015:

```
partition_utils -c /journal -o add -t realtime -s 07272015 -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.

Important: 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"](#).

Note: 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.

Important: 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.

Caution: 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"](#).

Caution: You can use the **-f** (force) parameter to remove partitions even if the objects are associated with open items, but this parameter must be used with care.

Important: 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 the **-s** (start date) parameter to specify the start of the date range for the objects to remove. The format is *MMDDYYYY*.
- Use the **-e** (end date) parameter 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 the **-c** (base storable class) parameter 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.

Caution: 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 the **-t** (type) parameter to remove real-time or delayed-event partitions. The default is to remove both real-time and delayed-event partitions.
- Use the **-f** parameter 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 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 removes the partitions with real-time events from July 20, 2004, to September 20, 2004:

```
partition_utils -o remove -s 07202004 -e 09202004 -t realtime
```

The following command removes real-time partitions for item table entries from July 20, 2004, to September 20, 2004:

```
partition_utils -o remove -s 07202004 -e 09202004 -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"](#).

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 purged. See ["Running the partition_utils Utility in Test Mode"](#).

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.

To purge objects without removing partitions, run the **partitions_utils** utility with the **purge** operation. The syntax is as follows:

```
partition_utils -o purge [-s start_date] -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.
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 exist.

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 must 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.

Caution:

- 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 Data](#)".

Converting Nonpartitioned Classes to Partitioned Classes

If you did not enable partitioning for one or more storable classes when you installed Oracle Communications Billing and Revenue Management (BRM), follow the procedures in this chapter to convert specified nonpartitioned storable classes to partitioned storable classes in the BRM database.

Before partitioning your storable classes, read the following:

- [Partitioning Tables](#)
- [About Purging Data](#)

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 "[Overview of Enabling Partitioning during Installation](#)".

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.

Tip: 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*).

Important:

- If you download to a Windows workstation, use **FTP** to copy the **.bin** file to a temporary directory on your UNIX server.
 - You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid “Out of Memory” error messages in the log file. See “Increasing Heap Size to Avoid ‘Out of Memory’ Error Messages” in *BRM Installation Guide*.
-
-

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.

Caution: You must source the **source.me** file to proceed with installation. Otherwise, “suitable JVM not found” and other error messages appear.

Bash shell:

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

C shell:

```
source source.me.csh
```

4. Go to *temp_dir* and run the following command:

```
7.5.0_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/7.5**.

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.

Important: 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. See ["Starting and Stopping the BRM System"](#).

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 27-1](#) 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.

Important: 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 27–1 Conversion Scripts and Files

Script or File	Description
crt_pinlog.sql	SQL script that creates the pinlog files.
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	Master 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.
<i>storable_class_name_tables_tobe_partitioned.sql</i>	Custom SQL scripts that implement partitioning for the specified storable class (<i>storable_class_name</i>). 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.

About Purging Data

This chapter provides information about purging obsolete database objects and expired account subbalances from your Oracle Communications Billing and Revenue Management (BRM) database. It discusses the impact of purging major event objects and how this affects BRM applications.

Before purging data, you should be familiar with the following:

- Basic BRM concepts. See "Introducing BRM" 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 28–1](#) 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 28–1 *Objects That Are Automatically Partitioned and Purged*

Object Type	Purging Criteria
bill	<ul style="list-style-type: none">■ Within specified date range■ Closed

Table 28–1 (Cont.) Objects That Are Automatically Partitioned and Purged

Object Type	Purging Criteria
event	<ul style="list-style-type: none"> Within specified date range Associated items are closed <p>Note: Event objects in the partition_historic partition cannot be purged even if they meet all the purging criteria.</p>
invoice	<ul style="list-style-type: none"> Within specified date range
item	<ul style="list-style-type: none"> Within specified date range Closed Not referenced by a journal object <p>Note: Item objects in the partition_historic partition cannot be purged even if they meet all the purging criteria</p>
journal	<ul style="list-style-type: none"> Within specified date range Fully earned by the end date of the date range <p>Note: In addition, Oracle recommends that the journal object be included in a G/L report.</p>
newsfeed	<ul style="list-style-type: none"> None
sepa	<ul style="list-style-type: none"> 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.

Caution: 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 with Event Browser.

Important: 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.

See "[Associating Items with Nonpurgeable Events](#)".

Important: 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](#)".

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 chapter.

Billing Event Objects

[Table 28–2](#) lists the impacted Billing Event Objects.

Table 28–2 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	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> 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. <p>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.</p>
/event/billing/cycle and all of its subclasses	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> 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. For information about the pin_ledger_report utility, see "pin_ledger_report" in BRM Collecting General Ledger Data. You cannot perform A/R activities such as adjustments, write-offs, disputes, and settlements for subscription charges logged by these events. <p>If you generate invoices and G/L reports, keep these event objects through the current billing cycle.</p>
/event/billing/cycle/discount/mostcalled	<p>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.</p> <p>See also the impact of purging /event/billing/cycle and all of its subclasses.</p>
/event/billing/cycle/rollover/monthly	<p>Purging these objects will have the following impact:</p> <ul style="list-style-type: none"> 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. <p>See also the impact of purging /event/billing/cycle and all of its subclasses.</p>
/event/billing/cycle/rollover_transfer	<p>If you purge these events, information about the balance impacts of the original rollover event will be lost.</p> <p>See also the impact of purging /event/billing/cycle and all of its subclasses.</p> <p>You may want to keep these objects for auditing purposes.</p>
/event/billing/deal	Abstract class, not persisted in the database.

Table 28–2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/deal/cancel	<p>If you purge these events:</p> <ul style="list-style-type: none"> ■ Rerating, which must replay these events, will not work correctly. However, you can purge old events that occurred before your rerating time frame. ■ Best pricing will not work correctly because rerating is called during best pricing. ■ You cannot see the corresponding event history in Customer Center. You can, however, see event history contained in other types of deal event objects that were not purged. Keep these event objects as long as you want to display deal cancellation history.
/event/billing/deal/purchase	<p>If you purge these events:</p> <ul style="list-style-type: none"> ■ Rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame. ■ Best pricing will not work correctly because rerating is called during best pricing. <p>You cannot see deal purchase history. Keep these event objects as long as you want to display the history of deal purchases.</p>
/event/billing/debit	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> ■ The Bill Details panel in Customer Center cannot display debit event objects associated with a particular item. ■ 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. <p>Keep these event objects as long as you want to do the following:</p> <ul style="list-style-type: none"> ■ Display debit events. ■ Display event details in an invoice. ■ Create custom reports that use debit events.
/event/billing/discount/action/cancel	<p>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.</p> <p>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</p>
/event/billing/discount/action/modify	<p>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.</p> <p>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.</p> <p>Keep these event objects if you must display the history of discount attribute and status modification.</p>

Table 28–2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/discount/action/modify/status	<p>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.</p> <p>Keep these event objects if you must display the history of discount attribute and status modification.</p>
/event/billing/discount/action/purchase	<p>If you purge these event objects, you cannot see the corresponding discount history.</p> <p>Keep these event objects as long as you want to display the history of discount purchases.</p>
/event/billing/discount/action/set_validity	<p>If you purge these events:</p> <ul style="list-style-type: none"> Rating will not consider the validity period setting because the discount validity period settings before 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	<p>If this event is purged, discount amount based on the number of active subscriptions will not be applied.</p>
/event/billing/mfuc_update	<p>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.</p>
/event/billing/ordered_balgrp	<p>Abstract class, not persisted in the database.</p>
/event/billing/ordered_balgrp/create	<p>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.</p>
/event/billing/ordered_balgrp/delete	<p>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.</p>
/event/billing/ordered_balgrp/modify	<p>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.</p>
/event/billing/product	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> 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. <p>Keep these event objects if either of the following is true:</p> <ul style="list-style-type: none"> You use Customer Center to view current or historical information for canceled and item products. Your custom applications use these event objects.
/event/billing/product/action/cancel	<p>If you purge these events:</p> <ul style="list-style-type: none"> Rerating, which must replay these events, will not work correctly. However, you can purge old events that occurred before your rerating time frame. Best pricing will not work correctly because rerating is called during best pricing. You cannot see the corresponding event history in Customer Center. You can, however, see event history contained in other types of deal event objects that were not purged. Keep these event objects as long as you want to display deal cancellation history.

Table 28–2 (Cont.) Billing Event Objects

Event object	Impact of purging/when to purge
/event/billing/product/action/modify /event/billing/product/action/modify/status	If you purge these events: <ul style="list-style-type: none"> ■ Rerating will not work correctly. However, you can purge old events that occurred before your rerating time frame. ■ Best pricing will not work correctly because rerating is called during best pricing. ■ 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 must display the history of product attribute and status modification.
/event/billing/product/action/purchase	If you purge these events: <ul style="list-style-type: none"> ■ 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.
/event/billing/product/action/set_validity	If you purge these events: <ul style="list-style-type: none"> ■ Rating will not consider the validity period setting because the product's validity period settings before first usage and the new calculated validity period after first usage will be lost. <p>Also during the back-out process, rerating uses these events to remove the validity setting.</p> <p>Note: You can purge old events that occurred before your rerating time frame.</p>
/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. <p>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.</p>
/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 28–3 lists the Accounts Receivable Event Objects.

Table 28–3 Accounts Receivable Event Objects

Event object	Impact of purging/when to purge
/event/billing/adjustment/account /event/billing/adjustment/item	<p>If you purge these event objects, you lose some G/L information.</p> <p>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.</p> <p>Keep these event objects for the specified G/L posting period.</p>
/event/billing/adjustment/event	<p>If you purge these event objects, you lose some G/L information.</p> <p>The pin_ledger_report utility uses these event objects to generate the G/L report for billed_earned and billed revenue types.</p> <p>Keep these event objects for the specified G/L posting period.</p>
/event/billing/adjustment/tax_event	<p>If you purge these event objects, you lose some G/L information.</p> <p>The pin_ledger_report utility uses these event objects to generate the G/L report for billed_earned and billed revenue types.</p> <p>Keep these event objects for the specified G/L posting period.</p>
/event/billing/batch/payment	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> ■ You cannot use the pin_recover utility with the resubmit parameter to resend failed transactions. ■ You cannot use the pin_clean utility. <p>You can purge the original batch payment event objects after you successfully resubmit any failed batch transactions.</p>
/event/billing/batch/reversal	<p>You can purge these event objects if you do not have any custom applications using them.</p>
/event/billing/dispute/item /event/billing/settlement/item	<p>If you purge these event objects, you lose some past G/L, dispute, and settlement information.</p> <p>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.</p> <p>Keep these event objects for the specified G/L posting period.</p>
/event/billing/item/transfer	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> ■ 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. <p>Keep transfer event objects as long as you keep their related billable items.</p>

Table 28–3 (Cont.) Accounts Receivable Event Objects

Event object	Impact of purging/when to purge
/event/billing/payment and all of its subclasses	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> ■ 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. <p>If you have already reversed payments or your company is bound by corporate policies prohibiting payment reversal after a specified period, you can purge these events.</p>
/event/billing/reversal and all of its subclasses	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> ■ 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. <p>Your business requirements determine how long you keep these event objects.</p>
/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	<p>If you purge these event objects, you lose some G/L and tax information.</p> <p>The pin_ledger_report utility uses these event objects to generate the G/L report for billed_earned and billed revenue types.</p> <p>Keep these event objects for the specified G/L posting period.</p>

Delayed Event Objects

Table 28–4 lists the Delayed Event Objects.

Table 28–4 Delayed Event Objects

Event object	Impact of purging/when to purge
event/delayed/tmp_profile_event_ordering	<p>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.</p>

Group Event Objects

Table 28–5 lists the Group Event Objects.

Table 28–5 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 28–6 lists the Sharing Group Event Objects.

Table 28–6 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 28–7 lists the Session Event Objects.

Table 28–7 Session Event Objects

Event object	Impact of purging/when to purge
/event/session and all of its subclasses	<p>If you purge these event objects:</p> <ul style="list-style-type: none"> 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. <p>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.</p> <p>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.</p>

Auditing Event Objects

Table 28–8 lists the Auditing Event Objects.

Table 28–8 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 "Using testnap" in *BRM Developer's Guide*.

 For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
6. Stop and restart the CM. For more information, see ["Starting and Stopping the BRM System"](#).

For more information about business profiles, see "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 ["pin_close_items"](#).

About Purging Account Subbalances

You can purge expired account subbalances that are no longer required for daily business operations from your BRM database. Expired subbalances are the validity-based balances whose valid end dates are in the past.

Caution: When you delete subbalances from the database, events that impacted those subbalances cannot be rerated. Make sure you no longer need the expired subbalances before deleting them.

You delete expired subbalances 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. Use the following command to run the **pin_sub_balance_cleanup** utility:

```
pin_sub_balance_cleanup -n | -d
```

For more information, see ["pin_sub_balance_cleanup"](#).

Generating Virtual Columns on Event Tables

This chapter describes 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* class and subclasses).

About Generating Virtual Columns on Event Tables

Oracle Database 11g enables 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 11g. 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 classes (*/event* and its subclasses) in the BRM schema to use virtual columns. You convert event 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 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.

- **PIN_FLD_BRAND_OBJ**

The **brand_obj_type** column is converted to a virtual column.

After the event 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 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 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 the discussion on creating custom applications in *BRM Developer's Guide* for information on how to make your custom applications support virtual columns.
- See the discussion on Rated Event (RE) Loader in *BRM System Administrator's Guide* for information on how to set up RE Loader when using a virtual column-enabled system.
- See the discussion on the Conversion Manager `pin_cmt` utility in *BRM System Administrator's Guide* for information on moving legacy data when using a virtual column-enabled system.

Generating Virtual Columns on Event Tables

This section describes how to generate virtual columns in the BRM database for event tables (for the `/event` 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 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:

- Install BRM 7.5 Patch Set 3.
See *BRM Patch Set 3 Installation Guide*.
- 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.
- (Optional) If you have a nonpartitioned event table and you want to enable partitioning, install the 7.5_PartitionUpg_ *platform_32_opt*.bin package and enable partitioning for the event tables.

Important: 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.

- (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 are 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 an error occurs before the job finishes 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.

Exporting a BRM Schema with Virtual Columns

After you generate virtual columns, if you must export your BRM schema, you must remove the virtual columns from the schema before the export. In addition, if you must restore the exported schema, you must 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 11g 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 must 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

Part VII 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](#)

Managing a Multischema System

This chapter provides guidelines to help you manage an Oracle Communications Billing and Revenue Management (BRM) multischema system.

Before you read this chapter, you should be familiar with how BRM works. See "BRM System Architecture" in *BRM Concepts*.

For more information about monitoring and maintaining a BRM system, see ["Monitoring and Maintaining Your BRM System"](#).

Important: Multischema functionality is an optional component, not part of base BRM. For information on installing a multischema system, see "Installing a Multischema System" in *BRM Installation Guide*.

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* and "Configuring the Invoice pin.conf for Multiple Database Schemas" in *BRM Designing and Generating Invoices*.

The following are the main administrative tasks for a multischema system:

- [Converting a Single-Schema System to a Multischema System](#)
- [Preparing to Manage a Multischema System](#)
- [Adding Database Schemas to a Multischema System](#)
- [Setting Database Schema Status](#)
- [Setting Database Schema Priorities](#)
- [Creating Custom Tables That Are Available to All Database Schemas](#)
- [Synchronizing Database Schema Data Dictionaries](#)
- [Synchronizing the Database Schema /uniqueness Objects](#)
- [Changing the Interval for Updating the Distribution Table](#)

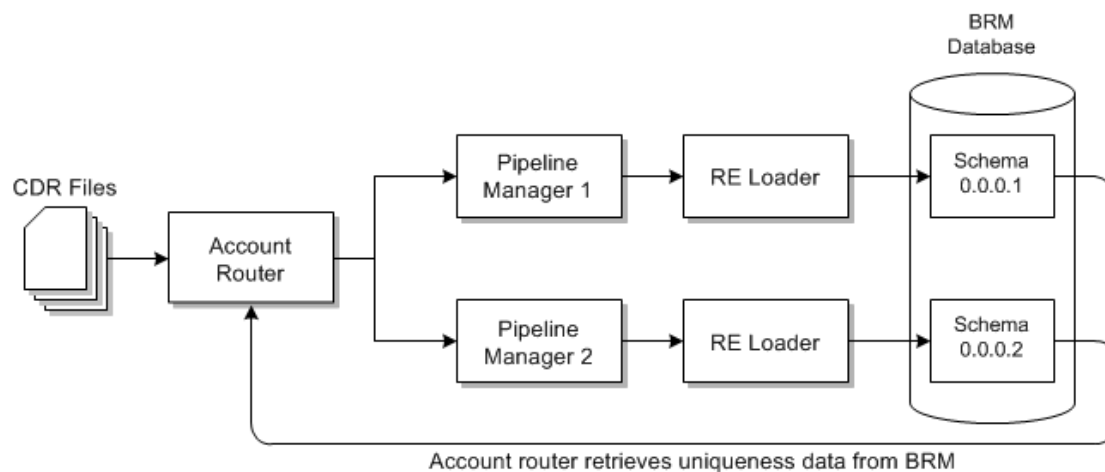
Using Pipeline Manager with Multiple Database Schemas

When you use Multidatabase Manager, you run an instance of Pipeline Manager for each BRM database schema. The following types of data must be managed:

- **Account data:** When account data changes in a database, the Account Synchronization Data Manager (DM) sends the data to Pipeline Manager. The DAT_Listener module map file specifies which Pipeline Manager instance the data is sent to.
- **CDR data:** Before a call details record (CDR) file is processed, the Pipeline Manager FCT_AccountRouter module separates CDRs by BRM database schemas and sends the CDRs to the appropriate Pipeline Manager instance.

Figure 30–1 shows how FCT_AccountRouter manages incoming CDRs:

Figure 30–1 FCT_AccountRouter CDR Management



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.

See "Installing the Third-Party Software" in *BRM Installation Guide*.

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 Multidatabase Manager on the Primary Installation Machine" in *BRM Installation Guide*.

4. Configure the BRM software by editing the **pin_setup.values** file.

Important: 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.

See "Installing the Third-Party Software" in *BRM Installation Guide*.

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 *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 more 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 an account's parent, branded, or 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, sponsoring, or branded 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.

Caution: 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" ;

DB_NO = "0.0.0.2" ;          # 2nd database config. block
PRIORITY = 3;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN" ;
```

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). See ["Starting and Stopping the BRM System"](#).

Tip: 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 of the same brand in the same schema
- All nonpaying (subordinate) child accounts in the same schema as their parent accounts
- All sponsored accounts in the same schema as their sponsoring accounts

To customize how schemas are selected, see "Creating Accounts in a Multischema System" in *BRM Managing Customers*.

Important: 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.

Caution: 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" ;

DB_NO = "0.0.0.2" ;           # 2nd database config. block
PRIORITY = 3 ;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "OPEN" ;

DB_NO = "0.0.0.3" ;           # 3rd database config. block
PRIORITY = 5 ;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "CLOSED" ;
```

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"](#).

Note: The **load_config_dist** utility requires a configuration file. See ["Creating Configuration Files for BRM Utilities"](#).

6. Stop and restart all CMs. See ["Starting and Stopping the BRM System"](#).

Tip: 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.

Note: This utility needs a configuration (**pin.conf**) file in the directory from which you run the utility. For information about creating configuration files for BRM utilities, see "[Creating Configuration Files for BRM Utilities](#)".

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. See "[Starting and Stopping the BRM System](#)".
5. Verify that the **/uniqueness** object was loaded by using one of the following to display the **/uniqueness** object:
 - Object Browser. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
 - **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 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. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
 - **robj** command with the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Multischema Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) multischema utilities.

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 cannot load separate **/config/distribution** objects for each brand. All brands use the same object.

Caution:

- 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.
-

Important: 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.

Important: You must restart the Connection Manager (CM) to make new resources 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.

Caution:

- The **pin_multidb.pl** script calls this utility to create a **/uniqueness** object from a **/service** object in a multischema environment; therefore, stop the **pin_multidb.pl** script 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.
-

Important: 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.

Important: You must restart the CM to make new resources 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.

Important: 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_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 31–1](#) lists the commands that you can enter at the monitor prompt.

Table 31–1 *pin_mta_monitor* Commands

Command	Description
p	Starts monitoring <i>mta_application</i> . The utility prints the thread activity for <i>mta_application</i> to stdout . See the discussion on multithreaded applications in <i>BRM System Administrator's Guide</i> .

Table 31–1 (Cont.) pin_mta_monitor Commands

Command	Description
t [+ -] <i>number</i>	Regulates the thread load on your MTA utility, where <i>number</i> specifies the number by which you want to increase or decrease the number of threads that <i>mta_application</i> uses. <ul style="list-style-type: none">▪ t [+] <i>number</i> adds the specified number of threads to the thread pool of <i>mta_application</i>.▪ t [-] <i>number</i> removes the specified number of threads from the thread pool of <i>mta_application</i>.
?	Displays the pin_mta_monitor utility's commands.
q	Stops monitoring <i>mta_application</i> 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.

Important:

- 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

Important: To fully configure a multischema system, run the script with **-i**, and then run it again with **-f**.

-i

Initializes the primary and secondary schemas.

-f

Finalizes the multischema installation.

-R all | 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 the discussion about 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 31–2](#):

Table 31–2 -R Group Values

-R Group Value	Description
CONFIG	Refreshes configuration objects
PRICE	Refreshes pricing objects
UNIQUENESS	Refreshes uniqueness objects
GENERAL	Refreshes 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 31–3](#):

Table 31–3 -r Group Values

-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.

-h

Displays the syntax and parameters for this script.

Results

If the **pin_multidb.pl** script does not notify you that it was successful, check that the data in both schemas is the same, or look in the log file (**pin_multidb.log**) to find any errors. The log file is either in the directory from which the script was started or in a directory specified in the configuration file.

Part VIII

Migrating Accounts

Part VIII describes how to migrate accounts from one database to another in an Oracle Communications Business and Revenue Management (BRM) system. It contains the following chapters:

- [Understanding Account Migration](#)
- [Installing and Configuring BRM for Account Migration](#)
- [Migrating Accounts with the Pipeline Manager Running](#)
- [Using Account Migration Manager](#)
- [Migrating Accounts within an Oracle IMDB Cache Grid](#)
- [Modifying Applications to Work with AMM](#)
- [Modifying the Account Migration Manager](#)
- [AMM Entity Relationship Diagram](#)
- [Account Migration Utilities](#)

Understanding Account Migration

This chapter provides an overview of account migration in Oracle Communications Billing and Revenue Management (BRM). It describes the Account Migration Manager (AMM) application, including how AMM interacts with your BRM database.

Before migrating accounts, you should be familiar with the following topics:

- BRM concepts. See "Introducing BRM" in *BRM Concepts*.
- Oracle In-Memory Database (IMDB) Cache Manager. See "[Managing IMDB Cache-Enabled Systems](#)".
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.

About Account Migration

Account migration consists of migrating (or transferring) the data associated with selected accounts from their source storage location to a destination storage location.

BRM migrates account data based on account migration information composed of details on the selected accounts and access information for the source and destination locations of those accounts. You provide this information to the utility you use to perform the migration.

When you store data only in the BRM database, the migration process moves the data associated with the selected accounts from the source database schema to a destination schema.

When you use Oracle IMDB Cache Manager in your system, this migration process includes the Oracle IMDB Cache associated with the source BRM database schema and the Oracle IMDB Cache associated with the destination BRM database schema. After the migration process ends, the destination BRM database schema holds the account information for the migrated accounts and the destination Oracle IMDB Cache holds the critical data for the migrated accounts.

When you use Oracle IMDB Cache Manager, you sometimes migrate account data from one Oracle IMDB Cache instance to another within the same Oracle IMDB Cache grid. In that case, the storage location of the account data in the BRM database remains unchanged. The migration process ensures that the database table entries are updated accordingly. After the migration, when you modify the cached data for an account in its new Oracle IMDB Cache location, the appropriate data in the BRM database is updated. The synchronization between the cached data in Oracle IMDB Cache and the corresponding data in the BRM database is maintained and no data is lost.

When to Migrate Accounts

You migrate account objects to redistribute the data in the following situations:

- One BRM database schema contains significantly more or fewer accounts than other BRM database schemas (for example, when you add a schema to an existing multischema system).
- The number of events per account is significantly greater in one schema than in other schemas.
- The time it takes to complete a billing run becomes erratic.
- A node is added to an Oracle IMDB Cache grid.

About Account Migration Processes

The process you choose for an account migration task depends on the current storage location of the account objects selected for migration. The possible scenarios are as follows:

- [Account Migration Involving Only the BRM Database](#)
- [Account Migration Involving a BRM Database and an Oracle IMDB Cache](#)
- [Account Migration Between the Logical Partitions in One IMDB Cache Grid](#)

Account Migration Involving Only the BRM Database

If you configured your BRM environment with only the BRM database, you use the AMM application to perform all account migration tasks. You provide information on the accounts to migrate and the access details for the source database schema and the destination schema. See "[About the AMM Application](#)".

Account Migration Involving a BRM Database and an Oracle IMDB Cache

If you configured your BRM environment with Oracle IMDB Cache Manager, the accounts selected for migration are stored in a BRM database schema and their critical data might be located in the Oracle IMDB Cache associated with that database schema.

This migration consists of the following steps:

1. You use the AMM application to perform account migration from the source BRM database schema to the destination BRM database schema. See "[How AMM Works with Oracle IMDB Cache Manager in the BRM Environment](#)".
2. After the application completes the migration, you store the critical data for the migrated accounts in the destination Oracle IMDB Cache by using the appropriate SQL script created and stored during the installation process. See "[Loading Destination Oracle IMDB Cache with Data from BRM Database](#)".

Account Migration Between the Logical Partitions in One IMDB Cache Grid

In this scenario, you are redistributing data between the Oracle IMDB Caches (logical partitions) within *one* Oracle IMDB Cache grid only.

You use the **pin_amt_tt** utility to move the accounts between the Oracle IMDB Caches in one Oracle IMDB Cache grid. The account data in the BRM database is not migrated. See "[Migrating Accounts within an Oracle IMDB Cache Grid](#)".

You can set up synchronization between Pipeline Manager and the **pin_amt_tt** utility by modifying the configuration file used by this utility. See "[About Account Synchronization with Pipeline Manager after the Migration](#)".

Scheduling Account Migration

You achieve optimal migration performance and the smallest impact to your operations if you schedule migrations for maintenance windows. If you must migrate a large number of accounts but your maintenance window is only a couple of hours, you can perform migrations over a period of several days. The AMM software processes jobs in stages, enabling you to pause and resume account migration without affecting database integrity.

For information on scheduling migration, see "[Automating Account Migration](#)".

About the AMM Application

AMM is an application that migrates accounts and all of their associated objects from a source schema in a BRM database to a destination schema in the same database.

How AMM Works

The AMM software migrates accounts based on information you provide in the account search configuration file for that migration task.

In this file, you specify a group of accounts to migrate based on specific criteria, such as account creation date or account status. The group of accounts and associated objects that meet this criteria forms a *job*.

Jobs are processed by the AMM software through a queue system, with each job processed in the order received. To improve migration performance, jobs are subdivided into *batches*, which contain a configurable number of accounts.

Batches are assigned to a configurable number of threads, which process the batches in parallel. Each batch is migrated in a single distributed transaction, during which time activity is prevented on the accounts in the batch. Depending on the success of the batch migration, changes to the database are either committed or rolled back.

How AMM Works with Oracle IMDB Cache Manager in the BRM Environment

When your environment includes the Oracle IMDB Cache Manager, data from a single database schema might be distributed among multiple logical partitions (also called Oracle IMDB Caches). As a result, accounts selected for migration might have data stored in one of the Oracle IMDB Caches associated with the source BRM database schema.

In addition to the information on the accounts to migrate, you provide the access details for the source Oracle IMDB Cache in the **Infranet.properties** configuration file.

Before the migration job starts, AMM locates the data in the source Oracle IMDB Cache (for the selected accounts), updates the corresponding data objects in the BRM database, and unloads the cache groups for the selected accounts from the source Oracle IMDB Cache. AMM migrates the selected account data from the source BRM database location to the destination BRM database location.

About Migrating Accounts When the Pipeline Manager Is Online

You can configure your system to migrate accounts when the Pipeline Manager is running. In this configuration, the Pipeline Manager temporarily stops call details record (CDR) processing for all accounts undergoing migration. See "[Migrating Accounts with the Pipeline Manager Running](#)".

By default, AMM does not support migration when your pipelines are running. You specify whether AMM can migrate accounts while the Pipeline Manager is online by using the **controller_N_event_generation** parameter in the AMM **Infranet.properties** file. See ["Connecting AMM to Your Database Schemas"](#).

Caution: If you disable this option, you must shut down the Pipeline Manager before you migrate accounts. Otherwise, your pipelines might send account information to the wrong BRM database schema, causing incorrect account balances and revenue leakage.

Deleting Migrated Objects

For performance reasons, the AMM software does not automatically remove the migrated objects from the source database schema. Instead, it flags the migrated objects as *invalid* to prevent BRM applications from accessing them. You can use the AMM software to manually purge invalid objects from the database at any time.

About Migrating Hierarchical, Sponsorship, and Resource Sharing Accounts

You can configure AMM to migrate hierarchical, sponsorship, and resource sharing groups from a source BRM database schema to a destination schema. In this configuration, AMM does the following:

- Searches for accounts in two phases. See ["About Searching for Member and Nongroup Member Accounts"](#).
- Organizes accounts that meet the search criteria by account group. See ["About Account Groups"](#).
- Migrates entire account groups. See ["About Migrating Account Groups"](#).

By default, group migration is disabled. You specify whether to migrate account groups by using the **migration_mode** entry in the account search file (*BRM_home/apps/amt/account_search.cfg*). See ["Creating the Account Search Configuration File"](#).

Caution: When you enable group migration, you must perform extra verification steps to prevent accounts from being severed from their associated account group. See ["Checking Account Group Details"](#).

About Searching for Member and Nongroup Member Accounts

When you enable group migration, AMM searches for accounts in two phases:

- **In the first phase, AMM searches for nongroup member accounts only.** That is, AMM excludes all remittance, sponsorship, hierarchical, branded, and resource sharing accounts from the account search. Accounts meeting the search criteria are divided into batches, and each batch is flagged as containing nongroup member accounts only.
- **In the second phase, AMM searches for accounts belonging to hierarchical, sponsorship, and resource sharing groups only.** If an account meets the search criteria, AMM finds all other account members that are related to the account. These accounts are organized into an *account group*. See ["About Account Groups"](#).

Each account group is divided into batches, which are assigned an account group ID and flagged as containing account group members.

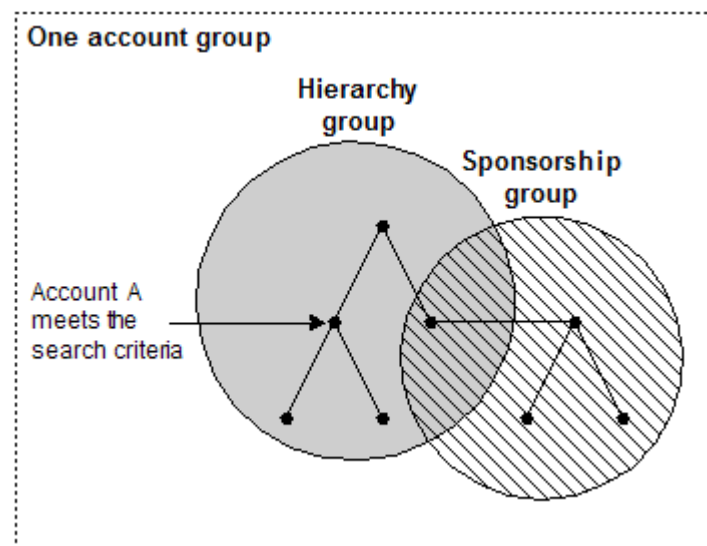
Note: AMM cannot migrate remittance or brand accounts. See ["Brand and Remittance Accounts Not Migrated"](#).

All accounts meeting the search criteria, both group member and nongroup member accounts, still form one job.

About Account Groups

An account group consists of all account members that are related to a specific account. When AMM finds an account that meets the search criteria, it finds the parent account and all other child accounts in the group. If one of the accounts is also a member of another group, it finds all members of the other group as well.

Figure 32-1 One Account Group



For example, in [Figure 32-1](#), account A meets the search criteria. Because account A is a child in a hierarchy group, AMM finds the parent account and all child accounts in that hierarchy group. Because one hierarchy account member is also a member of a sponsorship group, AMM finds all accounts in the sponsorship group as well. In this example, the account group consists of all accounts in the hierarchy group and the sponsorship group.

About Migrating Account Groups

AMM migrates account group member and nongroup member batches in different ways. AMM migrates:

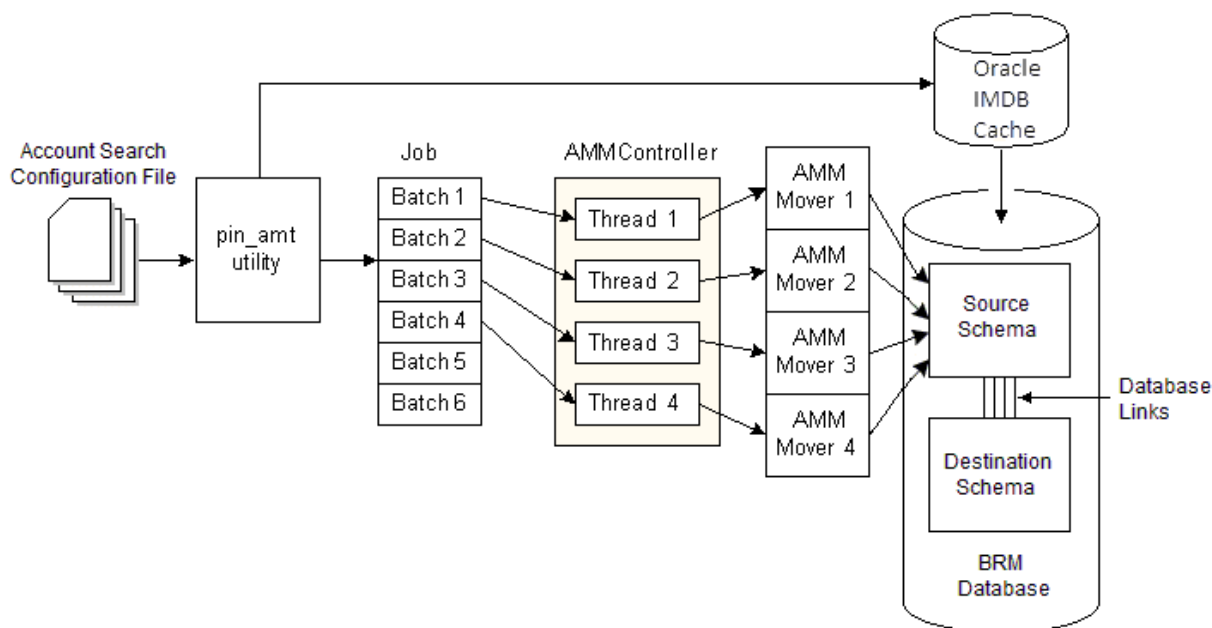
- **Batches containing nongroup members in one transaction.** During migration, AMM disables all accounts that belong to a single batch.
 - If the batch migrates successfully, AMM commits the changes to all database schemas and enables the accounts in the *destination* schema.
 - If the batch fails to migrate, AMM rolls back the changes and re-enables the accounts in the *source* database schema.

- **Batches containing account group members by account group ID.** AMM disables all accounts that belong to an account group before migration begins. After migration starts, AMM monitors whether all batches for the account group migrate successfully.
 - If all batches migrate successfully, AMM commits the changes to all database schemas and enables the accounts in the *destination* schema.
 - If even one batch in the group fails, AMM leaves all account group members disabled in both source and destination schemas. You must fix the error and use AMM to reprocess the job before your BRM system can access the accounts.

AMM Process Overview

Figure 32–2 shows the AMM process overview.

Figure 32–2 AMM Process Overview



The account migration process can be divided into the following stages:

1. The account search configuration file containing information on the accounts to migrate and access information for the source and destination database schemas is provided as input to the **pin_amt** utility. See ["About the Account Search Configuration File"](#).
2. The **pin_amt** utility processes the account search configuration file.
If accounts in the account search configuration file have data in the corresponding Oracle IMDB Cache grid, the **pin_amt** utility ensures that the data for those accounts is current in the BRM database.
3. Every account to be migrated is set up as an individual job. The jobs are divided into batches and placed in the queue. See ["About the pin_amt Utility"](#).
4. The AMM Controller allocates batches to threads and passes batches to the AMM Mover. See ["About the AMM Controller"](#).

5. The AMM Mover migrates the batch of accounts by moving the associated data from the source schema to the destination schema. See ["About the AMM Mover"](#).
6. If Oracle IMDB Cache Manager is used in the system, you update the cache groups in the logical partition of the destination Oracle IMDB Cache grid.

About the Account Search Configuration File

The account search configuration file provides the details on which accounts to migrate, their source and destination storage locations, and the size of each batch.

You can also migrate accounts based on custom criteria. See ["Creating Custom Account Search Criteria"](#).

About the pin_amt Utility

The **pin_amt** utility is a standalone utility which generates account migration jobs for the AMM Controller to process. This utility can perform all its functions, such as finding accounts to migrate and submitting and enabling jobs, whether the AMM Controller is online or offline.

You use the **pin_amt** utility to perform the following tasks:

- Start, stop, resume, and monitor the AMM Controller.
- Find all accounts in the source database schema that meet your search criteria.
- Enable account migration jobs in the queue.
- Delete jobs from the job management tables.
- Purge invalid objects from the source schema.

When you submit an account search configuration file, the **pin_amt** utility does the following:

1. Searches the source database schema for all accounts that meet the criteria in the account search configuration file, excluding all remittance accounts and all accounts that are part of branded, hierarchical, sponsorship, and resource sharing groups.

For accounts that have data in a corresponding Oracle IMDB Cache, this utility does the following:

- a. Accesses the account data in the source Oracle IMDB Cache.
 - b. Updates the data in the BRM database for those accounts with the data in the source cache.
 - c. Unloads the cache groups associated with those accounts from that Oracle IMDB Cache.
2. Divides the list of account POIDs into batches.
 3. Populates the job management tables in the primary, source, and destination schemas with the list of account POIDs to migrate. See ["About AMM Job Management Tables"](#).
 4. Determines whether group migration is enabled.
 - If group migration is *enabled*, **pin_amt** proceeds with steps 5 through 9.
 - If group migration is *disabled*, the account search is complete. The AMM Controller can begin processing the job when the job is enabled in the queue. See ["About the AMM Controller"](#).

5. Searches all hierarchy, sponsorship, and resource sharing groups in the source schema for accounts that meet the search criteria.
6. When an account meets the search criteria, **pin_amt** finds all other accounts related to the account and organizes them into an account group.
7. Determines whether the size of the account group exceeds the maximum. If it does, AMM excludes the account group from the job.

Note: You specify the maximum size of an account group by using the account search configuration file. See ["Creating the Account Search Configuration File"](#).

8. Divides all members of one account group into batches. All batches in the group are assigned the same group ID and flagged as containing account group members.
9. Populates the job management tables in the primary, source, and destination schemas with the list of account POIDs to migrate. See ["About AMM Job Management Tables"](#).

About the AMM Controller

The AMM Controller is a server process that checks the queue for jobs to process. By default, your system contains one AMM Controller, which processes one job at a time. For information about using multiple AMM Controllers, see ["About Using Multiple AMM Controllers"](#).

The AMM Controller processes group member and nongroup member batches in different ways:

- [How AMM Controller Processes Batches with Nongroup Members](#)
- [How AMM Controller Processes Batches with Account Group Members](#)

How AMM Controller Processes Batches with Nongroup Members

When processing batches that contain nongroup members, the AMM Controller does the following:

1. Assigns batches to threads, which run in parallel. Each thread processes one batch at a time. If there are more batches than threads, each thread must process multiple batches in sequence. You use a configuration file to configure the number of AMM Controller threads. See ["Connecting AMM to Your Database Schemas"](#).
2. Changes the batch status to IN_PROGRESS. See ["About Batch Status Flags"](#).
3. Passes the job ID, batch number, and schema qualifications name to the AMM Mover on the destination database schema for processing. See ["About the AMM Mover"](#).
4. Determines whether the AMM Mover successfully migrated accounts.
 - If migration is successful, the AMM Controller commits the changes to all database schemas and changes the batch status to FINISHED. See ["About Batch Status Flags"](#).
 - If migration fails, the AMM Controller rolls back the changes and updates the batch status to FAILED. See ["About Batch Status Flags"](#).

How AMM Controller Processes Batches with Account Group Members

When processing batches that contain account group members, the AMM Controller does the following:

1. Changes the account group status to GROUP_DISABLING. See ["About Group Status Flags"](#).
2. Locks the appropriate base table records in the source database schema so that applications cannot access group member accounts during migration.
3. Marks all account group members in the source schema as invalid.
4. Determines whether all account group members were disabled in the source schema.
 - If all accounts were successfully disabled, the AMM Controller changes the account group status to GROUP_READY. See ["About Group Status Flags"](#).
 - If any accounts were not disabled, the AMM Controller changes the account group status to FAILED. See ["About Group Status Flags"](#).
5. Changes the account group status to GROUP_IN_PROGRESS.
6. Passes individual batches in the account group to the AMM Mover for processing.
 - a. Assigns an individual batch in the group to a thread.
 - b. Changes the batch status to IN_PROGRESS.
 - c. Passes the job ID, batch number, and database link name to the AMM Mover. See ["About the AMM Mover"](#).
 - d. Determines whether the AMM Mover successfully migrated the batch and sets the batch status to FINISHED or FAILED.
7. Determines whether all batches in the account group migrated successfully.
 - If all batches migrated successfully, the AMM Controller enables all account group members in the destination database schema and changes the account group status to GROUP_FINISHED. See ["About Group Status Flags"](#).
 - If any batch failed to migrate, the AMM Controller changes the account group status to GROUP_FAILED. See ["About Group Status Flags"](#).

Important: When an account group fails to migrate, all its accounts remain disabled in the source and destination schemas. You must fix the error and migrate the job again before your BRM system can access the accounts.

About the AMM Mover

The AMM Mover is the process that actually moves accounts from one database schema to another. Each schema contains at least one AMM Mover.

Note: The AMM Mover performs the following functions regardless of whether a batch contains group-member or nongroup member accounts.

When the AMM Mover receives a batch, it does the following:

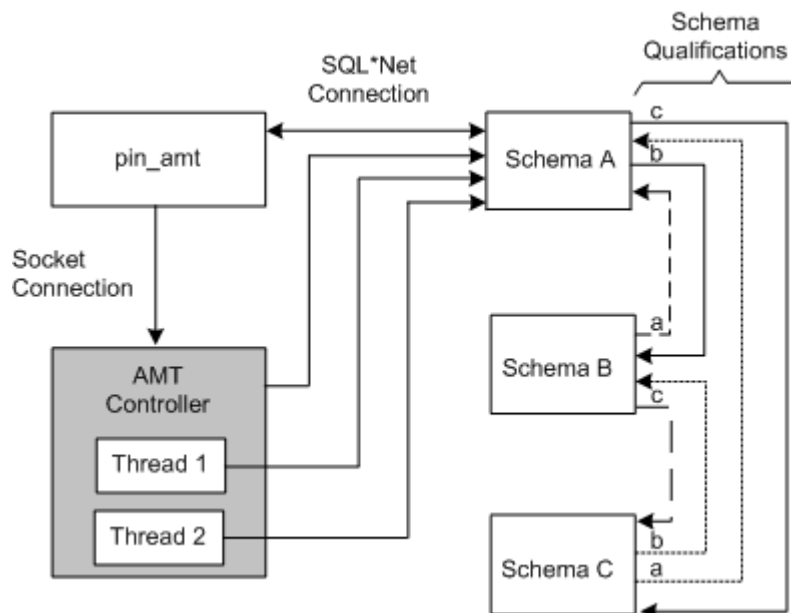
1. Locks the appropriate base table records in the source database schema so that applications cannot access accounts during migration.
2. Migrates the objects for an account batch from the source schema to the destination schema in a single distributed transaction. See "[About Distributed Transactions](#)".
3. Marks all migrated objects in the source schema as invalid.
4. Updates the account POIDs in the uniqueness table to reflect the account's new location. For example, if an account is migrated from schema **0.0.0.1** to schema **0.0.0.2**, the account POID changes from **0.0.0.1 /account 2286 0** to **0.0.0.2 /account 2286 0**.

About Distributed Transactions

The AMM software migrates each batch of accounts as a single distributed transaction by using schema qualifications. Hence, changes can be made to the primary, source, and destination database schemas and then committed or rolled back in one transaction, ensuring the integrity of the database.

[Figure 32–3](#) shows the schema qualifications for a multischema system with three database schemas, one AMM Controller, and two threads:

Figure 32–3 AMM Database Schema Connections



Account Migration Restrictions

When you run the AMM software, be aware of the following restrictions:

- [Account Activity Prevented during Account Migration](#)
- [Do Not Rerate Events during Account Migration](#)
- [Do Not Alter Account Group Members](#)
- [Migration Prevented during Account Activity](#)
- [Unique POIDs Required across All Database Schemas](#)

- [Transient Objects Are Not Migrated](#)
- [Some Client Applications May Fail during Account Migration](#)
- [AMM Does Not Support Some BRM Components](#)
- [Using BRM Reports after Migration](#)

Account Activity Prevented during Account Migration

To prevent applications from accessing or modifying accounts that are being migrated, AMM locks the accounts in Oracle. Only one batch of accounts per thread is locked at a time and only while the accounts are being physically migrated.

Note: When migrating account groups, AMM locks all accounts in the account group before migration begins.

If an application attempts to access a locked account, the Oracle Data Manager (DM) returns a PIN_ERR_INVALID_OBJECT error.

Do Not Rerate Events during Account Migration

Because the AMM software may suspend some events that you want to rerate, you *must not* rerate pipeline events during account migration.

Do Not Alter Account Group Members

AMM checks account group relationships only when you first create a job, and does not reverify relationships during the migration process. Therefore, once an account group is included in a migration job, you:

- *Must not* add members to the account group
- *Must not* modify relationships between account group members

If you must alter an account group *after* it is included in a job but *before* the job completes migration, you must:

1. Delete the migration job.
2. Modify the account group.
3. Re-create the account migration job.

Migration Prevented during Account Activity

AMM does not migrate accounts while they are being accessed or modified by BRM or another application. For best performance, stop all account activity before you migrate accounts.

If you cannot restrict all access to the accounts in your database schemas, AMM can still process account migration jobs. However, AMM does not migrate any batch that contains active accounts. You can check for failed batches and resubmit them for migration once account activity stops.

Brand and Remittance Accounts Not Migrated

You can migrate accounts based on a variety of criteria, including account status, account creation date, and product name. However, the AMM software always excludes brand and remittance account from account migration jobs.

Unique POIDs Required across All Database Schemas

The AMM software can migrate only accounts that have a unique POID. Starting with Infranet Release 6.2 ServicePak 1, the multischema software automatically creates unique POIDs across all database schemas in your system.

If your database contains accounts created both before and after you installed Release 6.2 ServicePak 1, AMM migrates only those accounts created *after* you installed 6.2 ServicePak 1. For information on how to migrate accounts created before 6.2 ServicePak 1, contact your Oracle BRM representative.

Important: If your BRM system uses a custom POID generation scheme, make sure the sequence number generation algorithm creates unique POIDs across all of your database schemas.

Transient Objects Are Not Migrated

While the `pin_amt` utility migrates account objects, it does not migrate any transient objects.

Some Client Applications May Fail during Account Migration

BRM client applications, such as Customer Center, Payment Tool, and Self-Care Manager, may generate error messages and fail to commit changes to the database during account migration. For example, if a CSR opens an account in Customer Center just before the account being migrated to another database schema, Customer Center generates an “Unable to process account” and “ERR_INVALID_OBJECT” error message when the CSR attempts to save any changes to the account.

In that case, the CSR must restart Customer Center and access the account again so it retrieves the account’s new location.

Important: If your system contains custom client applications that connect to the BRM database and search accounts based on POID, you must modify your application. See ["Modifying Custom Client Applications for AMM"](#).

AMM Does Not Support Some BRM Components

Currently, you cannot use AMM to migrate accounts if your BRM system includes any of the following optional components:

- LDAP Manager
- RADIUS Manager

Using BRM Reports after Migration

To use BRM Reports after you migrate accounts, you must use BRM Reports Release 6.2 ServicePak 1 or later. If you use an earlier version of BRM Reports with AMM, your

reports will retrieve and process duplicate data from your source and destination database schemas.

For example, if an account object is migrated from schema **0.0.0.1** to schema **0.0.0.2**, earlier versions of BRM Reports retrieve the account object from both schemas while BRM Reports 6.2 ServicePak 1 and later retrieve the account object only from schema **0.0.0.2**.

For information on how to modify custom reports to work with AMM, see ["Modifying Custom BRM Reports for AMM"](#).

About Using Multiple AMM Controllers

Caution: Implementing multiple AMM Controllers is for advanced users only. Use multiple AMM Controllers only if you understand the impact to migration performance.

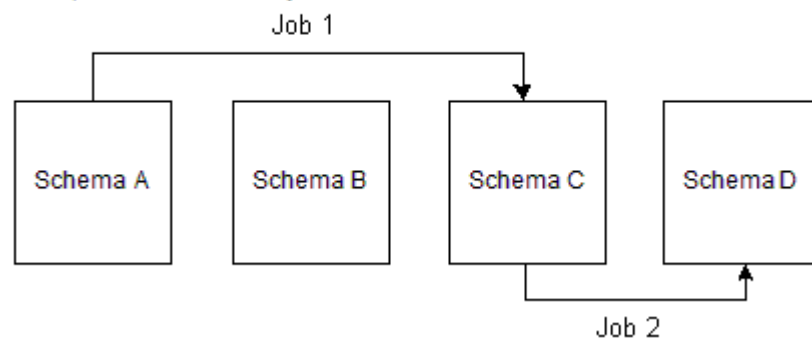
Using multiple AMM Controllers enables you to process multiple account migration jobs in parallel. However, you receive performance improvements only in the following situations:

- Your system contains more than three database schemas.
- No two migration jobs use the same schema at the same time.

When multiple jobs use the same schema, as shown in [Figure 32–4](#), migration performance degrades significantly.

Figure 32–4 Concurrent Database Use Performance Degradation

Job 1 and Job 2 use database schema C at the same time, which decreases performance.



For more information, contact your Oracle BRM representative.

Account Migration Performance

Account migration is resource intensive and can overload your BRM system.

Signs that your system is overloaded during account migration:

- Batch processing times steadily increase, without returning to their initial processing times.
- The AMM software is processing fewer than five accounts per second.

- You receive a distributed transaction time-out error (Oracle error 2049).
- There are a high number of waits for undo segment extension and latch free operations.

If your system exhibits any of these signs, you must tune your Oracle database. For guidelines, see ["Tuning Your Database for Optimal Account Migration Performance"](#) or contact your Oracle BRM representative.

About AMM Job Management Tables

Job management tables are created on your database schemas during installation and are populated with information about each migration job.

The AMM installer creates the tables listed in [Table 32–1](#) on your schemas:

Table 32–1 *AMM Job Management Tables*

Table Name	Schema	Description	When Populated
AMT_ACCOUNT_BATCH_TABLE_T	Primary only	Stores the list of tables containing data to migrate for a particular batch.	Populated by the AMM Mover when it migrates a particular batch.
AMT_METADATA_T	All schemas	AMM data dictionary. This lists all default BRM tables and any custom tables you created. If you add any tables after you install AMM, you must refresh the AMM data dictionary. See "Configuring AMM for New Custom Tables" .	During installation and when you run pin_amt_install.pl -m to refresh the AMM data dictionary.
AMT_POID_TYPE_MAP_T	All schemas	Maps the POID type to the table name. This table is static.	During installation and when you run pin_amt_install.pl -m to refresh the data dictionary.

About Job Status Flags

The AMM software sets jobs to a status listed in [Table 32–2](#). You can see a job's status by running a **list_jobs** report. See ["Monitoring Job Status"](#).

Table 32–2 *Job Status Flags*

Status	Description
DISABLED	The job has been submitted but not enabled.
NOT_PROCESSED	The account migration job is enabled and waiting in the queue to be processed.
PRE_MIGRATION_WAITING	AMM is notifying the account-router Pipeline Manager to suspend events for all accounts in the job.
PRE_MIGRATION	The account-router Pipeline Manager acknowledged that it is suspending events for all accounts in the job. AMM is waiting a specified amount of time before starting migration.
READY	The job is ready to be processed.

Table 32–2 (Cont.) Job Status Flags

Status	Description
IN_PROGRESS	The job is being processed by the AMM Controller.
FAILED	The job has been aborted.
BAL_MIGRATED	The account discount balance migrated successfully.
POST_MIGRATION_WAITING	AMM is notifying the account-router, source, and destination instances of the Pipeline Manager that the job migrated successfully.
POST_MIGRATION	The account-router, source, and destination instances of the Pipeline Manager acknowledged that they updated all account information in their caches.
PRE_JOB_RECYCLE	AMM is notifying the account-router Pipeline Manager to resume processing events for accounts in the job.
JOB_RECYCLE	The account-router Pipeline Manager acknowledged that it is ready to begin reprocessing events for accounts in the job.
FINISHED	The job has completed successfully.

About Batch Status Flags

AMM sets account batches to a status listed in [Table 32–3](#). You can check a batch's status by running a **job_details** report. See "[Checking Job Details](#)".

Table 32–3 Batch Status Flags

Status	Description
NOT_PROCESSED	The account batch has not yet been migrated.
IN_PROGRESS	The account batch is currently being migrated.
FAILED	The account batch failed to migrate. All changes to the database have been rolled back.
FINISHED	The account batch migrated successfully. All changes to the database have been committed.

About Group Status Flags

AMM sets account groups to a status listed in [Table 32–4](#). You can check an account group's status by running a **group_details** report. See "[Checking Account Group Details](#)".

Table 32–4 Group Status Flags

Status	Description
NOT_PROCESSED	The account group has not yet been migrated.
GROUP_DISABLING	All account group members are being disabled in the source database schema. That is, AMM is marking all account group members as invalid to prevent applications from accessing those accounts.
FAILED	AMM did not disable all account group members in the source schema.
GROUP_READY	All account group members were successfully disabled in the source schema. AMM can begin processing batches.
GROUP_IN_PROGRESS	The account group is currently being migrated.
GROUP_FAILED	The account group failed to migrate to the destination schema.
GROUP_FINISHED	The account group migrated successfully.

Installing and Configuring BRM for Account Migration

This chapter explains how to install and configure software required for account migration with the Oracle Billing and Revenue Management (BRM).

Before you read this chapter, you should be familiar with BRM, multischema, and Account Migration Manager (AMM) concepts and architecture. See the following topics:

- "Introducing BRM" in *BRM Concepts*
- "BRM System Architecture" in *BRM Concepts*
- "Putting Together Your BRM System" in *BRM Installation Guide*
- [Understanding Account Migration](#)

System Requirements

The system requirements for account migration consists of the following:

- [General Software Requirements](#)
- [Software Requirements for Oracle IMDB Data Manager](#)
- [Software Requirements for Account Migration Manager](#)

General Software Requirements

Before installing AMM, you must install:

- **Oracle 10g or Oracle 11g database software.** You must also install the following Oracle 10g or Oracle 11g components:
 - JServer
 - PL/SQL
 - SQL*Plus
- **Third-Party software**, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- **BRM.** See "Installing BRM" in *BRM Installation Guide*.
- **The latest BRM patch set.** See *Patch Set Installation Guide*.

- **Multidatabase Manager.** See "Installing a Multischema System" in *BRM Installation Guide*.

Pipeline Manager Requirements

If you plan to migrate accounts when the Pipeline Manager is running, you must also install the following BRM software:

- **Pipeline Manager.** See "Installing Pipeline Manager" in *BRM Installation Guide*.
- **Account Synchronization Data Manager (DM).** See "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.
- **Suspense Manager.** See "Installing Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
- **Rated Event (RE) Loader.** See "Installing Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

Important: These components include changes that enable you to migrate accounts while your pipelines are running. You must install the latest versions of these components to migrate accounts.

Software Requirements for Oracle IMDB Data Manager

If you have installed Oracle IMDB Data Manager, you must install:

- Oracle TimesTen 32-bit client software required to run the **pin_amt** utility:
 - TimesTen 11.2.2.3.0 client (compatible with Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache 11g Release 2)
 - TimesTen 11.2.1.9.0 client (compatible with Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache 11g Release 1)

For information on installing Oracle TimesTen 32-bit client software, see *Oracle TimesTen In-Memory Database Installation Guide*.

- Oracle TimesTen 64-bit client software required to run the **pin_amt_tt** utility:
 - TimesTen 11.2.2.3.0 client (compatible with Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache 11g Release 2)
 - TimesTen 11.2.1.9.0 client (compatible with Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache 11g Release 1)

For information on installing Oracle TimesTen 64-bit client software, see *Oracle TimesTen In-Memory Database Installation Guide*.

Software Requirements for Account Migration Manager

AMM is available for Oracle databases and the HP-UX IA64, Linux, AIX, and Solaris operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

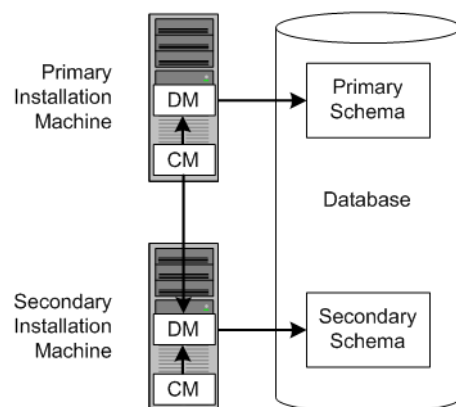
If you plan to use AMM software for account migration, you must also install:

- Java Development Kit (JDK 6)

Installing AMM

The instructions in this section assume that you have two BRM installation machines and two database schemas in your multischema environment as shown in [Figure 33–1](#).

Figure 33–1 Installing AMM



Installing AMM involves the following general steps:

1. [Configuring All Primary and Secondary Database Schemas](#)
2. [Installing the AMM Software on the Primary Installation Machine](#)
3. [Configuring Your Oracle DM to Check for Invalid Objects](#)
4. [Connecting AMM to Your Database Schemas](#)
5. [Configuring the TimesTen JDBC Driver Jar file for BRM](#)
6. [Configuring the `load_pin_uniqueness` Utility for Oracle IMDB Cache](#)

Configuring All Primary and Secondary Database Schemas

Before you can install AMM, you must configure your primary and secondary database schemas for account migration.

First, ensure that you assigned unique database instance names to each schema in your system:

1. Open the `tnsnames.ora` file in a text editor.
2. If necessary, assign a unique database instance name to each schema. For example, for the first schema:

```
Alias1 = (DESCRIPTION = (ADDRESS = (PROTOCOL=
TCP)(Host=DatabaseHostName)(Port= 1521))
(CONNECT_DATA = (SID =DatabaseSID)) )
```

For the second schema:

```
Alias2 = (DESCRIPTION = (ADDRESS = (PROTOCOL=
TCP)(Host=DatabaseHostName)(Port= 1521)) (CONNECT_DATA = (SID =DatabaseSID)) )
```

3. Save and close the file.

Then perform the following steps on *each* database schema in your multischema system:

1. Using SQL*Plus, log in to the database schema as the SYSTEM user and grant database linking privileges to the BRM user **pin**:

```
% sqlplus system/manager@databaseAlias
```

```
SQL> grant create database link to pin;
```

Grant succeeded.

2. Verify that JServer is installed on your system:

```
SQL> SELECT object_name, object_type FROM all_objects WHERE
       object_type = 'PACKAGE' and object_name = 'DBMS_JAVA';
```

If JServer is installed on your system, you receive the following:

OBJECT_NAME	OBJECT_TYPE
DBMS_JAVA	PACKAGE

If JServer is not installed, you receive the following:

no rows selected

3. Install JServer if it is not already installed on your system:
 - a. Add the following entry to the Oracle **initSID.ora** file (**\$ORACLE_HOME/dbs/initSID.ora**) of the schema's database instance:


```
java_pool_size=20971520
```
 - b. Restart Oracle so that the schema's database instance is initialized with your changes.
 - c. Install JServer manually by running the Oracle **initjvm** script:

```
% sqlplus sys/change_on_install@databaseAlias
```

```
SQL> @$ORACLE_HOME/javavm/install/initjvm.sql
```

For information, see your Oracle documentation.

4. Modify the entries listed in [Table 33–1](#) in the Oracle **initSID.ora** file (**\$ORACLE_HOME/dbs/initSID.ora**) of the schema's database instance:

Table 33–1 *initSID.ora Entries*

Entry	Description
global_names	Specifies whether a database link is required to have the same name as the database schema to which it connects. Set this to False .
utl_file_dir	Specifies the location of the Oracle utl_file . Set this to a writable directory for the user oracle .

5. Restart Oracle so that the schema's database instance is initialized with your changes.

Installing the AMM Software on the Primary Installation Machine

Note: If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install AMM, perform the following steps on the primary installation machine:

1. Log in as user **pin** and make sure the following environment variables listed in [Table 33–2](#) are set correctly in the **.cshrc** or **.profile** file:

Table 33–2 AMM Environment Variables

Environment Variable	Description
PATH	Make sure this includes the path to SQL*Plus and Xterm.
DISPLAY	Set this to <i>WorkstationName:0.0</i> . Note: You must set this variable to have the AMM software open an Xterm window that displays the AMM Controller's status in real time.
PIN_HOME	Specifies the directory where BRM is installed. The default is /opt/portal/7.5 .
ORACLE_HOME	Specifies the directory where Oracle is installed.

2. Download the software to a temporary directory (*temp_dir*).

Important:

- If you download to a Windows workstation, use **FTP** to copy the **.bin** file to a temporary directory on your UNIX server.
 - You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. See "Increasing Heap Size to Avoid 'Out of Memory' Error Messages" in *BRM Installation Guide*.
-

3. Go to the directory where you installed the Third-Party package and source the **source.me** file.

Caution: You must source the **source.me** file to proceed with installation, otherwise "suitable JVM not found" and other error messages appear.

Bash shell:

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

C shell:

```
source source.me.csh
```

4. Go to the *temp_dir* directory and enter this command:

```
7.5.0_AccountMigrationMgr_platform_32_opt.bin
```

where *platform* is the operating system name.

Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the **DISPLAY** environment variable before you install the software.

5. Follow the instructions displayed during installation. The default installation directory for AMM is **opt/portal/7.5**.

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

6. Go to the directory where you installed the AMM package and source the **source.me** file:

Bash shell:

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

C shell:

```
source source.me.csh
```

7. Go to the *BRM_home/setup* directory and run the **pin_setup** script.
8. Verify that the *BRM_home/setup/scripts/pin_multidb.conf* file contains accurate information about each database schema in your system. The **pin_amt_install** script uses the information in this file to set up your AMM environment.
9. For optimal performance, store your AMM job management tables and indexes in their own physical tablespaces. Map the following entries to four separate physical tablespaces by modifying the *BRM_home/setup/scripts/pin_tables.values* file:

```
$PIN_CONF_TBLSPACE14  
$PIN_CONF_TBLSPACE15  
$PIN_CONF_TBLSPACE16  
$PIN_CONF_TBLSPACE17
```

See "Database Configuration and Tuning" in *BRM Installation Guide*.

10. Log in as user **pin**, go to the *BRM_home/setup/scripts* directory, and run the **pin_amt_install** script:

```
# su - pin  
% cd BRM_home/setup/scripts  
% perl pin_amt_install.pl
```

11. Verify that installation was successful by checking the AMM installation log file (*BRM_home/setup/scripts/pin_amt_install.log*).
12. Restart your BRM processes. For information, see ["Starting and Stopping the BRM System"](#).

Configuring Your Oracle DM to Check for Invalid Objects

During account migration, the AMM software marks all migrated objects in the source database schema as invalid. To prevent BRM applications from accessing these objects, you must configure your Oracle Data Manager (DM) to check for invalid objects during account searches.

Caution: If you do not make this change, BRM applications retrieve duplicate data from your source and destination database schemas. For example, if an account is migrated from schema **0.0.0.1** to schema **0.0.0.2**, the application retrieves the account object from both schemas.

To configure your system to check for invalid objects, perform the following on *every* machine containing an Oracle DM:

1. Add the following line to the *BRM_home/sys/dm_oracle/pin.conf* file:

```
- dm dm_nul_poid_db_chk 1
```
2. Restart **dm_oracle** so that your Oracle DM is initialized with your changes. See ["Starting and Stopping the BRM System"](#).

Connecting AMM to Your Database Schemas

During installation, the AMM installer generates an **Infranet.properties** configuration file that specifies how to connect to your database schemas and the number and configuration of your AMM Controllers. The installer populates the connection parameters in the **Infranet.properties** file with values from your multischema **pin_multidb.conf** file and provides default values for setting up one AMM Controller. Use the **Infranet.properties** file to provide the information necessary for AMM to access the data for the accounts in your system.

Configuring the AMM Infranet.properties File

To configure the AMM **Infranet.properties** file, perform the following on the primary installation machine:

1. Go to the *BRM_home/sys/amt* directory and open the **Infranet.properties** file in a text editor. *BRM_home* is the directory in which you installed BRM components.

Note: The content of the **Infranet.properties** file conforms to the Java properties file conventions. Options are key-value pairs separated by the equal sign (=). For example, **0.0.0.1_user_name=pin** and **0.0.0.1_primary=true**.

2. Verify that the configuration entries reflect your environment. When checking the **Infranet.properties** file, make sure it contains the following:
 - Accurate information for connecting to all database schemas in your system. See ["Configuring Database and AMM Mover Information for Multischema Systems"](#).
 - Accurate information on AMM Controllers. See ["Configuring AMM Controller Definitions"](#).
 - If IMDB Cache is in use in your environment:

- Accurate information has been provided about the Oracle IMDB Cache associated with the source BRM database schema. See ["Specifying Schema Alias Information for Multischema Systems"](#).
- The unloading of cache groups from the source Oracle IMDB Cache is enabled. See ["Enabling the Unloading of Cache Groups from the Source Oracle IMDB Cache"](#).
- Access parameter values for each logical partition in the source cache have been provided. See ["Specifying Access Information for the Source Oracle IMDB Cache"](#).

3. Save and close the file.

Configuring Database and AMM Mover Information for Multischema Systems

Verify that the **AMM Infranet.properties** file contains a set of **0.0.0.x** entries for each database schema in your system. For example, if you have three database schemas in your system, the file should contain a set of entries prefixed by **0.0.0.1**, a set prefixed by **0.0.0.2**, and a set prefixed by **0.0.0.3**. [Example 33–1](#) shows the entries for a system with two database schemas.

Verify that the **Infranet.properties** file contains information on the AMM Mover associated with each database schema. If a schema contains more than one AMM Mover, make sure to update the **Infranet.properties** file accordingly.

Example 33–1 Example Configuration Information for Multischema Systems

```
#
# primary database schema
#
0.0.0.1_user_name=pin57204
0.0.0.1_user_password=&aes|08|0D5E11BFDD97D2769D9B0DBFBD1BBF7EED4E2DD0A43B0DE9BBE19D80ECB2FCF27A
0.0.0.1_instance_name=futt11
0.0.0.1_primary=true
# AMT mover
0.0.0.1_mover_log_file_dir=unknown
0.0.0.1_mover_log_file_flag=N
0.0.0.1_grp_srch_log_file_dir=unknown
0.0.0.1_grp_srch_log_file_flag=N
#
# secondary database schema
#
0.0.0.2_user_name=pin57204m2
0.0.0.2_user_password=&aes|10|0D5E11BFDD97D2769D9B0DBFBD1BBF7EE481DCC10889074E6ADE3DC873FEA13819
0.0.0.2_instance_name=futt11m2
0.0.0.2_primary=false
# AMT mover
0.0.0.2_mover_log_file_dir=unknown
0.0.0.2_mover_log_file_flag=N
0.0.0.2_grp_srch_log_file_dir=unknown
0.0.0.2_grp_srch_log_file_flag=N
#
```

Configuring AMM Controller Definitions

The installer creates a set of **Controller_1** entries for setting up one AMM Controller in the **AMM Infranet.properties** file and populates them with default values, as shown in [Example 33–2](#).

Example 33–2 AMM Controller Definitions

```
# controller definitions
#
controller_1_log_directory=/export/pin7204/opt/portal/7.5/apps/amt
controller_1_port_number=18566
controller_1_server=slc00ghm
controller_1_thread_count=2
controller_1_syslog_priority=7
controller_1_event_generation=false
controller_1_concurrent_job_number=20
controller_1_hold_period=120
controller_1_amt_queue_owner_name=ping
controller_1_amt_queue_name=ifw_sync_queue_amt
#
```

Access the **Infranet.properties** file and verify that the entries are as required.

If you require more than one AMM Controller, determine the optimal number of AMM Controllers for your system. Make sure that you provide the optimal number of threads for each AMM Controller in your system. See ["About Using Multiple AMM Controllers"](#) for more information.

For the additional controllers, create a set of **Controller_2** entries, **Controller_3** entries, and so on in the **Infranet.properties** file.

Specifying Schema Alias Information for Multischema Systems

When you use multischema systems, the database layer of your BRM system consists of one primary schema and one or more secondary schemas in a single database. As a result, the schema alias is the same for both schemas.

Access the AMM **Infranet.properties** file and enter an alias for the secondary schema in the file.

[Example 33–3](#) shows the entries for a multischema system:

Example 33–3 Example Alias Information for Multischema System

```
#
# Connection entries for the Primary Database Schema
0.0.0.1_user_name=pin
0.0.0.1_user_password=pin
0.0.0.1_instance_name=Schema1Alias
0.0.0.1_primary=true
0.0.0.1_mover_log_file_dir=./mover/log
0.0.0.1_mover_log_file_flag=y

# Connection entries for the Secondary Database Schema
0.0.0.2_user_name=pin
0.0.0.2_user_password=pin
0.0.0.2_instance_name=Schema2Alias
0.0.0.2_primary=false
0.0.0.2_mover_log_file_dir=./mover/log
0.0.0.2_mover_log_file_flag=y
```

Enabling the Unloading of Cache Groups from the Source Oracle IMDB Cache

This verification is necessary if you use Oracle IMDB Cache Manager in your system.

Verify that the **timesten_enabled** parameter is set to **true** in the AMM **Infranet.properties** file as shown below:

```
#Parameter to indicate if timesten is used or not.
timesten_enabled=true
```

This setting enables the **pin_amt** utility to unload the cache groups associated with accounts selected for migration from the source Oracle IMDB Cache.

Specifying Access Information for the Source Oracle IMDB Cache

If you use an Oracle IMDB Cache Manager, specify the access parameter for the source Oracle IMDB Cache. If the accounts are migrating from more than one Oracle IMDB Cache, make sure you specify the access parameter values for each source Oracle IMDB Cache involved in the migration.

The AMM **Infranet.properties** file must contain valid entries for each of the following parameters:

- **timesten_node_<Node Index>_user**
- **timesten_node_<Node Index>_pwd**
- **timesten_node_<Node Index>_dsn**
- **timesten_node_<Node Index>_db_id**

where **<Node Index>** identifies the logical partition where the accounts currently reside. [Table 33–3](#) provides a description of the parameters.

[Example 33–4](#) shows the access parameters defined when accounts are migrating from two logical partitions in a cache grid.

Example 33–4 Access Parameter Definitions for Two Logical Partitions

```
timesten_node_1_user=pin57204
timesten_node_1_pwd=pin57204
timesten_node_1_dsn=tt_lp1
timesten_node_1_db_id=0.0.0.1
```

```
timesten_node_2_user=pin57204
timesten_node_2_pwd=pin57204
timesten_node_2_dsn=tt_lp2
timesten_node_2_db_id=0.1.0.1
```

AMM Infranet.properties File Parameters

[Table 33–3](#) shows the parameters used in defining the AMM **Infranet.properties** file.

Table 33–3 AMM Infranet.properties Values

Entry	Value	Description
0.0.0.x_user_name	String	Specifies the Oracle user name for the specified database schema.
0.0.0.x_user_password	String	Specifies the Oracle user password for the specified database schema.
0.0.0.x_instance_name	String	Specifies the SQL*Net database alias name you assigned in the tnsnames.ora file.

Table 33–3 (Cont.) AMM Infranet.properties Values

Entry	Value	Description
0.0.0.x_primary	true or false	Flag that indicates whether the database schema is the primary schema. For the primary schema, set this to true . For all secondary schemas, set this to false .
0.0.0.x_mover_log_file_dir	Path name	Specifies the directory of the AMM Mover log file on the specified database schema. Important: This path must match the path specified in the utl_file_dir entry of the initSID.ora file.
0.0.0.x_mover_log_file_flag	Y or N	Specifies whether you want the AMM Mover to create log files on the specified database schema.
0.0.0.x_grp_srch_log_file_flag	Y or N	Specifies whether you want AMM to create a log file for the account group stored procedure. Note: The stored procedure finds all account group members related to a specific account.
0.0.0.x_grp_srch_log_file_dir	Path name	Specifies the directory for the account group stored procedure log file.
controller_N_log_directory	Path name	Specifies the directory in which to create the AMM Controller log file.
controller_N_port_number	Integer > 1024	Specifies the TCP/IP port number for the connection between the pin_amt utility and the AMM Controller. Each AMM Controller instance requires a unique port number.
controller_N_server	String	Specifies the host name of the machine that runs the AMM Controller.
controller_N_thread_count	Positive integer	Specifies the number of AMM Controller processing threads. For optimal performance, the number of AMM Controller threads should be 1 to 2 times the number of CPUs on the destination schema that are dedicated to AMM.
controller_N_syslog_priority	1 (low) through 7 (high)	AMM Controller log message priority threshold. Messages with a lower priority are suppressed.
pin_amt_log_directory	Path name	Specifies the path to the pin_amt log file.
controller_N_event_generation	true or false	Specifies whether the AMM Controller migrates accounts when your pipelines are running. This configures AMM to notify Pipeline Manager when accounts are migrated. The default is false .
controller_N_concurrent_job_number	Positive integer	Specifies how many jobs the AMM Controller starts concurrently. The default is 20 . Note: This entry is required only if the controller_N_event_generation entry is set to Y .
controller_N_hold_period	Positive integer	Specifies how long the AMM Controller waits, in minutes, before it starts to migrate accounts. This provides time for your pipelines to flush any EDRs targeted for accounts that are being migrated. The default is 120 . Note: This entry is required only if the controller_N_event_generation entry is set to Y .
controller_N_amt_queue_owner_name	String	Specifies the user that created the acknowledgment queue. Note: This entry is required only if the controller_N_event_generation entry is set to Y .

Table 33–3 (Cont.) AMM Infranet.properties Values

Entry	Value	Description
controller_N_amt_queue_name	String	Specifies the name of the acknowledgment queue. The AMM Controller dequeues pipeline acknowledgment events from this queue. Note: This entry is required only if the controller_N_event_generation entry is set to Y .
infranet.login.type	1 or 0	Specifies whether AMM requires a user name and password to log in to the Connection Manager (CM). <ul style="list-style-type: none">1 specifies that AMM must provide a user name and password.0 specifies that AMM uses a “trusted” login that comes through a CM Proxy, for example, and does not require a user name and password in the properties file. Note: This entry is required only if the controller_N_event_generation entry is set to Y .
infranet.connection	String	Specifies the full URL for connecting to the primary CM. <ul style="list-style-type: none">For a type 1 login, the URL must include a user name and password. You must specify the service name and service POID (“1”), but the CM determines the database number. For example: pcp://root.0.0.0.1:password@hostname:12009/service/pcm_clientFor a type 0 login, the URL requires a full POID, including the database number. Note: This entry is required only if the controller_N_event_generation entry is set to Y .
publish_migrated_objects	String	Specifies the storable classes whose objects are stored in the MIGRATED_OBJECTS_T cross-reference table. You can list multiple storable classes by using a comma (,) as a delimiter. For example: /service,/billinfo,/payinfo. Note: This entry is required only if you integrate AMM with your external application using Oracle Application Integration Architecture (Oracle AIA) in a multischema environment.
timesten_enabled	true or false	Specifies whether Oracle IMDB Cache is used in the system. When set to true , account migration involves unloading and loading of cache groups from the cache grid before and after migration of accounts. The default is false .
timesten_node_ <Node Index>_db_id	String	The database number for the logical partition of the Oracle IMDB Cache grid.
timesten_node_ <Node Index>_dsn	String	The data store name for the logical partition of the Oracle IMDB Cache grid.
timesten_node_ <Node Index>_user	String	The user name required to log in to the data store associated with the logical partition of the Oracle IMDB Cache grid.
timesten_node_ <Node Index>_pwd	String	The password required to log in to the data store associated with the logical partition of the Oracle IMDB Cache grid.

Sample Infranet.properties File

The following sample **Infranet.properties** file is for a system with the following components:

- Two database schemas, each with an AMM Mover
- One AMM Controller
- Pipeline migration enabled
- Oracle IMDB Cache enabled, two logical partitions in the cache grid, access information for the cache

Example 33–5 Sample Infranet.properties File

```
#
# primary database schema
#
0.0.0.1_user_name=pin57204
0.0.0.1_user_password=&aes|08|0D5E11BFDD97D2769D9B0DBFBD1BBF7EED4E2DD0A43B0DE9BBE19D80ECB2FCF27A
0.0.0.1_instance_name=futt11
0.0.0.1_primary=true
# AMT mover
0.0.0.1_mover_log_file_dir=unknown
0.0.0.1_mover_log_file_flag=N
0.0.0.1_grp_srch_log_file_dir=unknown
0.0.0.1_grp_srch_log_file_flag=N
#
# secondary database schema
#
0.0.0.2_user_name=pin57204m2
0.0.0.2_user_password=&aes|10|0D5E11BFDD97D2769D9B0DBFBD1BBF7EE481DCC10889074E6ADE3DC873FEA13819
0.0.0.2_instance_name=futt11m2
0.0.0.2_primary=false
# AMT mover
0.0.0.2_mover_log_file_dir=unknown
0.0.0.2_mover_log_file_flag=N
0.0.0.2_grp_srch_log_file_dir=unknown
0.0.0.2_grp_srch_log_file_flag=N
#
# controller definitions
#
controller_1_log_directory=/export/pin7204/opt/portal/7.5/apps/amt
controller_1_port_number=18566
controller_1_server=slc00ghm
controller_1_thread_count=2
controller_1_syslog_priority=7
controller_1_event_generation=false
controller_1_concurrent_job_number=20
controller_1_hold_period=120
controller_1_amt_queue_owner_name=pinq
controller_1_amt_queue_name=ifw_sync_queue_amt
#
# pin_amt definitions
#
pin_amt_log_directory=/export/pin7204/opt/portal/7.5/apps/amt
#
# CM connection properties
# connect string in the format
# infranet.connection=pcp://LOGIN:PASSWORD@HOSTNAME:PORT/service/pcm_client
#   infranet.login.type=1
#
infranet.connection=pcp://root.0.0.0.1:&aes|08|0D5E11BFDD97D2769D9B0DBFBD1BBF7E5D40C305EDF3D77DF111
AAB8F781E92122@slc00ghm:12204/service/pcm_client
infranet.login.type=1
```

```
#
# timesten configuration
#

#Parameter to indicate if timesten is used or not.
timesten_enabled=true

#Parameter to indicate the PIN HOME of target node.
target_pin_home=/export/pin7204/opt/portal/7.5

# The next set of parameters are for the credentials of timesten
# datastores on the source side that would be involved in migration.
# The next 4 parameters need to be replicated for each data stores incrementing the number in the
parameter name
#
# DataStore USER, PASSWORD, DataStore Name, Database ID for the data store
#

oracle_db_pwd=pin57204

timesten_node_1_user=pin57204
timesten_node_1_pwd=pin57204
timesten_node_1_dsn=tt_lp1
timesten_node_1_db_id=0.0.0.1

timesten_node_2_user=pin57204
timesten_node_2_pwd=pin57204
timesten_node_2_dsn=tt_lp2
timesten_node_2_db_id=0.1.0.1
```

Configuring the TimesTen JDBC Driver Jar file for BRM

Complete this step if you use Oracle IMDB cache in your system. If you have more than one instance of BRM, you must complete this step for each instance.

To configure the TimesTen JDBC Driver Jar file for BRM:

1. Go to the *BRM_home/apps/amt* directory.
2. Copy the **ttjdbc6.jar** file from the *TimesTen32_home/lib* directory, where *TimesTen32_home* is the directory in which you installed the 32-bit TimesTen client application.

Configuring the load_pin_uniqueness Utility for Oracle IMDB Cache

To verify/configure the **load_pin_uniqueness** utility for migrating account data:

1. Open the **load_pin_uniqueness** utility configuration file (*BRM_home/apps/multi_db/pin.conf*) in a text editor.
2. Verify that the following **is_timesten** entry is present. Add the entry, if necessary:

```
- load_pin_uniqueness is_timesten 1
```

3. Verify that the following `per_schema_node_info` entry is present. Add the entry, if necessary.

```
- load_pin_uniqueness per_schema_node_info DB_NO:DB_NO
```

where:

`DB_NO:DB_NO` is the per-schema logical partition information.

For example:

If the schema `0.0.0.1` has three logical partitions (`0.0.0.1`, `0.1.0.1`, and `0.2.0.1`), the entry is:

```
- load_pin_uniqueness per_schema_node_info 0.0.0.1:0.1.0.1:0.2.0.1
```

4. Save and close the file.

What's Next?

AMM installation is now complete.

To migrate accounts while the Pipeline Manager is online, you must perform additional configuration steps. See ["Configuring Your System to Migrate Accounts When the Pipeline Manager Is Running"](#).

If your system does not use batch rating or you plan to shut down Pipeline Manager during account migration, you can begin migrating accounts. See ["Using Account Migration Manager"](#).

Configuring AMM for Additional Database Schemas

You must reconfigure the AMM software whenever you add a database schema to an existing multischema system.

To configure AMM for additional database schemas, perform the following procedure on the primary installation machine:

1. Delete all existing account migration jobs in the queue:

```
% pin_amt -d JobID
```

2. Log in as user `pin` and run the `pin_amt_install.pl` script:

```
# su - pin
% cd BRM_home/setup/scripts
% perl pin_amt_install.pl
```

This script reinstalls the job management tables on your database schemas.

Configuring AMM for New Custom Tables

AMM migrates data to and from the tables listed in the AMM data dictionary. This list includes all BRM tables and any custom tables that were on your system when you installed AMM. If you add any tables after you install AMM, you must update the AMM data dictionary.

To update the AMM data dictionary, perform the following on your primary installation machine:

1. Log in as user `pin` and go to the `BRM_home/setup/scripts` directory:

```
% su - pin
% cd BRM_home/setup/scripts
```

2. Run the **pin_amt_install.pl** script with the **-m** parameter:

```
% perl pin_amt_install.pl -m
```

This script updates the AMM data dictionary tables (AMT_META_DATA_T and AMT_POID_TYPE_MAP_T) on all database schemas in your system.

Tuning Your Database for Optimal Account Migration Performance

To tune your database for optimal account migration performance:

- Use cost-based optimization.
- Set the number of Oracle rollback segments to approximately two times the number of AMM Controller threads. Multiple rollback segments enable Oracle to automatically allocate one rollback segment for each transaction.
- Set the **transactions_per_rollback_segment** parameter in the **\$ORACLE_HOME/dbs/initSID.ora** file to a small number. For best results, set it to 1 or 2.
- Set the initial size for the rollback segment to twice the total data volume of the account batch. You can estimate the account batch data volume by multiplying the average number of events per batch with the batch size.

Tip: Use the EVENT_T and major child tables, such as EVENT_BAL_IMPACTS_T, to determine the average number of events per batch. Typically, 90% of the account batch data volume can be attributed to event data.

- Set the optimal size for the rollback segments to twice the initial size.
- Set the next size for the rollback segments to half the initial size.
- Set the maximum number of extends to unlimited.

For more information, see your Oracle documentation. For information on additional ways to tune your database for AMM, contact your Oracle BRM representative.

Migrating Accounts with the Pipeline Manager Running

This chapter explains:

- How BRM migrates accounts when Pipeline Manager is running.
- How to configure your Oracle Communications Billing and Revenue Management (BRM) system to migrate accounts when Pipeline Manager is running.

Before you read this chapter, you should be familiar with account migration and its configuration in BRM. See the following documents:

- [Understanding Account Migration](#)
- [Installing and Configuring BRM for Account Migration](#)

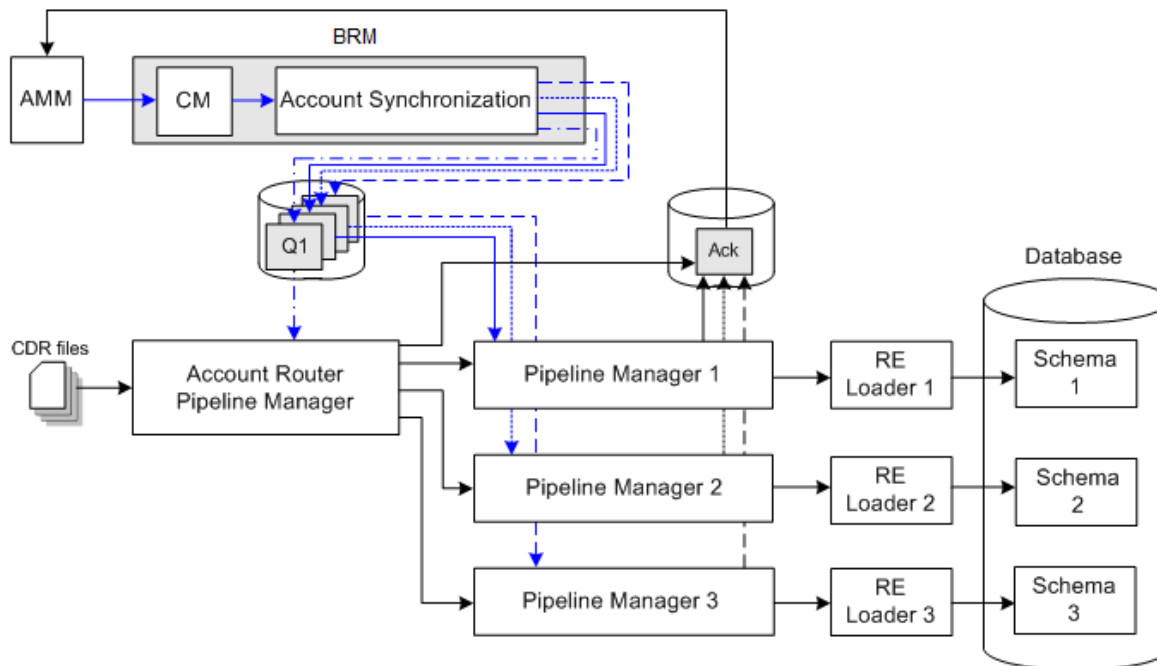
About Migrating Accounts When Pipeline Manager Is Online

BRM migrates accounts while the Pipeline Manager is running whether you use the AMM process (with its **pin_amt** utility) or the **pin_amt_tt** utility.

When you migrate accounts while Pipeline Manager is online, your pipelines stop processing any EDRs that apply to accounts undergoing migration. Your pipelines continue processing all other EDRs.

Important: When you migrate accounts within the same Oracle IMDB Cache grid, BRM synchronizes the pipeline for the destination BRM database schema based on whether you enabled synchronization in the **pin.conf** file used by the **pin_amt_tt** utility. See "[About Account Synchronization with Pipeline Manager after the Migration](#)".

[Figure 34–1](#) shows AMM interaction with the Pipeline Manager.

Figure 34–1 AMM Pipeline EDR Management

To coordinate account migration with your pipelines:

- AMM notifies the pipelines about a job's migration status by sending business events. See ["About Notifying the Pipelines about Account Migration"](#).
- The pipelines notify AMM about EDR processing status by sending acknowledgment events. See ["About Notifying AMM about EDR Processing"](#).
- The account-router Pipeline Manager suspends, recycles, and routes EDRs. See ["About the Account Router Instance of the Pipeline Manager"](#).

How AMM Interacts with Your Pipelines during Account Migration

The following steps outline how AMM interacts with your pipelines when processing account migration jobs:

1. AMM fetches a configurable number of migration jobs. See ["About Starting Multiple Jobs Concurrently"](#).
2. AMM notifies the account-router Pipeline Manager to hold EDRs for all accounts in a job.
3. The account-router Pipeline Manager begins holding all EDRs for the specified list of accounts and sends an acknowledgment to AMM. See ["About Suspending Call Records"](#).
4. AMM waits a specified amount of time before migrating accounts. See ["About Waiting before Migrating Accounts"](#).
5. AMM migrates all accounts in the job. See ["About the AMM Controller"](#).
6. AMM determines whether the job migrated successfully.
 - If migration finished successfully, AMM notifies the account router, source, and destination instances of the Pipeline Manager.

Note: If configured to do so, AMM also notifies any external applications.

- If migration failed, AMM does not send any notification to your pipelines and job processing stops.

Important: When migration fails, your pipelines continue to suspend all EDRs for the specified accounts. You must fix the problem and remigrate the job before the pipeline can begin reprocessing suspended EDRs.

7. The account router, source, and destination instances of the Pipeline Manager update their account information and send an acknowledgment to AMM.
8. AMM notifies the account router to resume processing EDRs for the specified list of accounts.
9. The account router resumes processing EDRs for the specified accounts and sends an acknowledgment to AMM.
10. AMM calls the PCM_OP_SEARCH_RECYCLE opcode to recycle suspended EDRs through the pipeline. See ["About Reprocessing Suspended Call Records"](#).

About Waiting before Migrating Accounts

After the account-router Pipeline Manager begins suspending EDRs, AMM waits a configurable amount of time before migrating a job. This provides time for your pipelines to flush any EDRs targeted for accounts in the migration job.

The default wait time is 120 minutes. You specify how long the AMM Controller waits before migrating accounts by using the **Controller_N_hold_period** entry in the AMM **Infranet.properties** file. For information, see ["Connecting AMM to Your Database Schemas"](#).

About Starting Multiple Jobs Concurrently

You can minimize the amount of time AMM spends in the waiting period by configuring AMM to start multiple migration jobs concurrently. In this configuration, AMM:

1. Fetches a configurable number of jobs.
2. Notifies the account-router Pipeline Manager to hold EDRs for multiple jobs.
3. Starts the timer for each job.
4. Once the waiting period is over, AMM migrates jobs individually.

This increases the number of jobs in the queue that are ready to be migrated.

You specify how many jobs an AMM Controller processes concurrently by using the **Controller_N_concurrent_job_number** entry in the AMM **Infranet.properties** file. See ["Connecting AMM to Your Database Schemas"](#).

About Notifying the Pipelines about Account Migration

AMM notifies your pipelines about account migration by sending a series of business events through the Account Synchronization architecture.

About AMM Business Events

AMM generates the five business events listed in [Table 34–1](#) to notify the account router, source, and destination instances of the Pipeline Manager when account migration occurs:

Table 34–1 *AMM Business Events*

Event	Recipient	Description
HoldCDRProcessing	Account-Router Pipeline Manager	Notifies the account-router Pipeline Manager to suspend all EDRs for a specified list of accounts.
ResumeCDRProcessing	Account-Router Pipeline Manager	Notifies the account-router Pipeline Manager to resume processing all suspended and new EDRs for the specified list of accounts.
MigrateAcct	Account-Router Pipeline Manager External Applications	Notifies the account-router Pipeline Manager and any external applications to update the account database location for the specified list of accounts.
MigrateSource	Source Pipeline Manager	Notifies the source Pipeline Manager that all accounts in the job migrated successfully.
MigrateDestination	Destination Pipeline Manager	Notifies the destination Pipeline Manager that all accounts in the job migrated successfully. The destination pipeline then reads account information from the database.

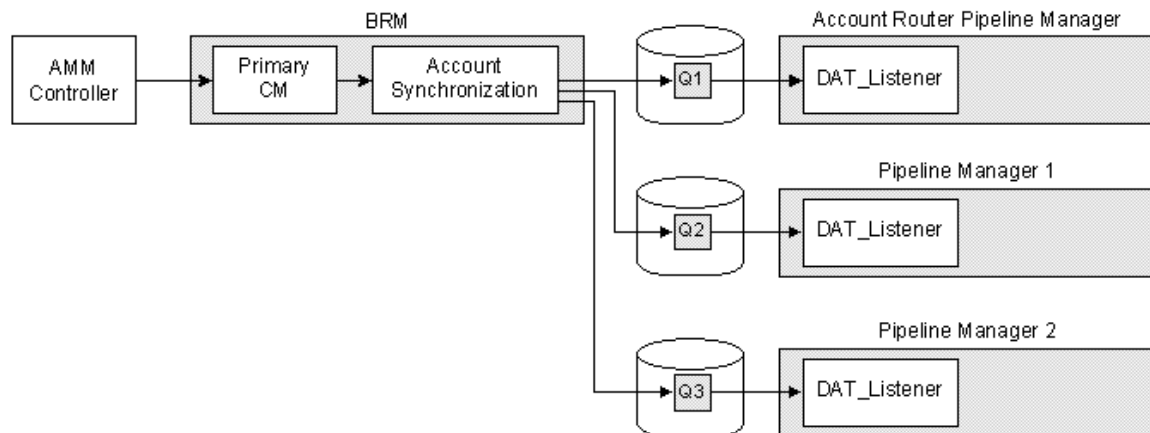
About Sending AMM Business Events to the Pipelines

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

Note: If configured to do so, AMM also sends business events to a queue for external applications.

AMM sends business events to a Pipeline Manager as follows:

1. AMM sends an event to the primary Connection Manager (CM).
2. The primary CM sends the event to the Account Synchronization architecture.
3. The Account Synchronization architecture uses its `ifw_sync_queue`names file to publish the business event to the appropriate queue.
4. The Pipeline Manager's DAT_Listener module dequeues the event and then forwards it to the appropriate pipeline data module.

Figure 34–2 Sending AMM Business Events

You configure your system to send AMM business events to your pipelines by:

- Connecting AMM to the primary CM.
- Creating an Oracle database queue for each instance of the Pipeline Manager.
- Configuring Account Synchronization to publish AMM business events to your queues.
- Configuring each instance of the Pipeline Manager to dequeue AMM business events from its associated Account Synchronization queue.

Figure 34–2 shows the AMM business events process described. See ["Configuring AMM to Send Business Events to Your Pipelines"](#).

About Notifying AMM about EDR Processing

Your pipelines notify AMM when it begins holding EDRs, reprocessing EDRs, or updating account data by sending acknowledgment events through a dedicated acknowledgment queue.

About Acknowledgment Events

Each instance of the Pipeline Manager generates acknowledgment events when the following actions listed in [Table 34–2](#) occur:

Table 34–2 Pipeline Manager Acknowledgment Events

Pipeline Instance	Sends Acknowledgments When...
Account router Pipeline Manager	<ul style="list-style-type: none"> ■ It begins suspending EDRs for a specified migration job. ■ It resumes processing EDRs for a specified migration job. ■ It completes an update of account locations in pipeline memory. This occurs after a migration job is successfully completed.
Pipeline Managers connected to the BRM database	<ul style="list-style-type: none"> ■ It completes an update of account locations in pipeline memory. This occurs when accounts in a job are successfully migrated to its associated database schema. ■ It receives a MigrateSource event from AMM to indicate that accounts were successfully migrated away from its associated database schema.

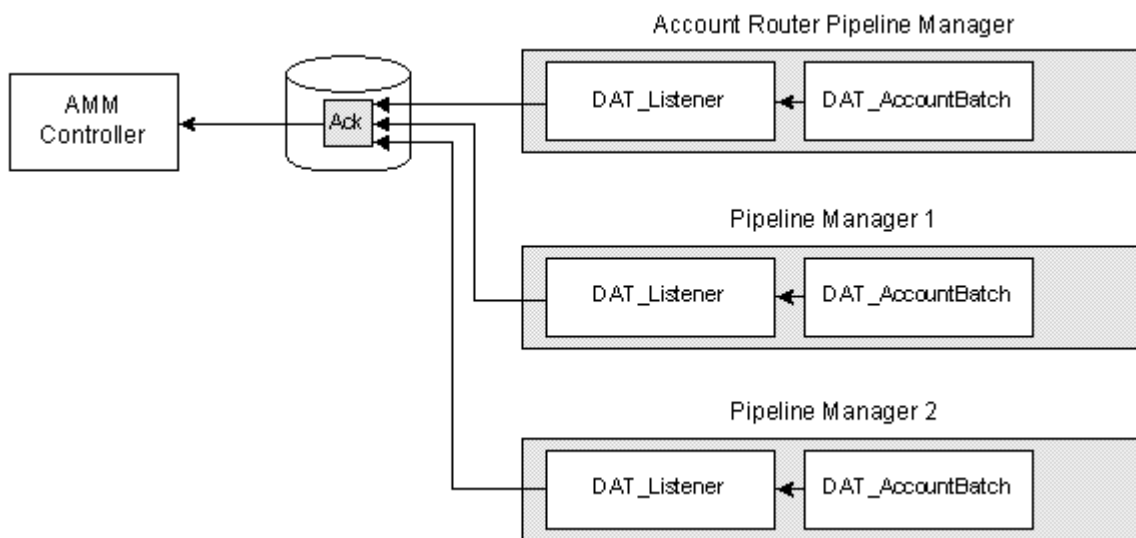
About Sending Acknowledgments to AMM

Each instance of the Pipeline Manager sends acknowledgment events to AMM by using a dedicated acknowledgment queue as shown in [Figure 34–3](#).

Each Pipeline Manager sends acknowledgments as follows:

1. The DAT_AccountBatch module sends an acknowledgment event to the DAT_Listener module.
2. The DAT_Listener module publishes the event to the acknowledgment queue.
3. The AMM Controller dequeues the event.

Figure 34–3 Sending Pipeline Acknowledgment Events to AMM



To configure your pipelines to send acknowledgments to AMM, you must:

- Configure the DAT_Listener module in *each* instance of the Pipeline Manager to publish acknowledgment events.
- Create a single database queue that is dedicated to acknowledgment events. All instances of the Pipeline Manager use this single queue.
- Configure the AMM Controller to dequeue events from the acknowledgment queue.

For more information, see ["Configuring Your Pipelines to Send Acknowledgments to AMM"](#).

About the Account Router Instance of the Pipeline Manager

The account router instance of the Pipeline Manager is used in multischema systems to route EDRs to the correct instance of the Pipeline Manager. For example, EDRs targeted for accounts that reside in database schema 3 are routed to the Pipeline Manager instance associated with schema 3.

When configured for migration, the account-router Pipeline Manager also performs the following tasks:

- Suspends EDRs targeted for accounts that are undergoing migration. See ["About Suspending Call Records"](#).

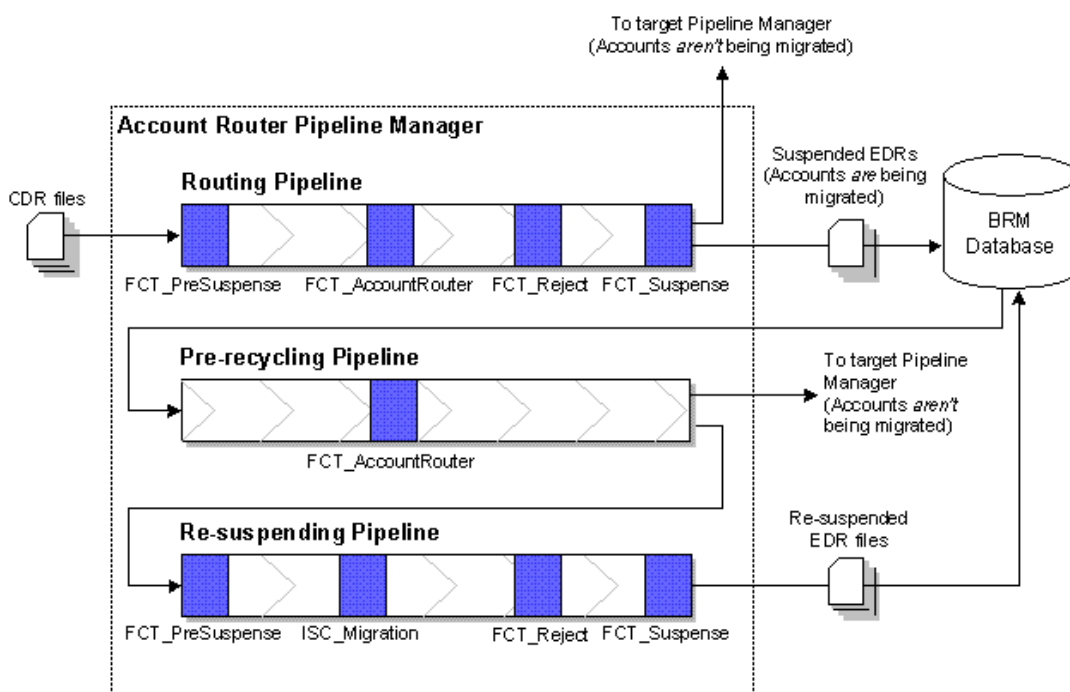
- Recycles previously suspended EDRs. See ["About Reprocessing Suspended Call Records"](#).

You configure the account-router Pipeline Manager for migration by creating three separate pipelines:

- A *routing pipeline* that routes EDRs to the appropriate instance of the Pipeline Manager and suspends any EDRs targeted for accounts undergoing migration. This pipeline must include the FCT_AccountRouter module, set to router mode; the FCT_PreSuspend module; the FCT_Reject module; and the FCT_Suspend module.
- A *pre-recycling pipeline* that processes previously suspended EDRs, determines whether an EDR is targeted for an account undergoing migration, and then routes the EDR to the appropriate output stream. This pipeline must include the FCT_AccountRouter module, set to recycle mode.
- A *re-suspending pipeline* that automatically suspends all EDRs. This pipeline must include the FCT_PreSuspend module, the ISC_Migration iScript, the FCT_Reject module, and the FCT_Suspend module.

You must also configure the account router data pool to pass migration status information to your pipelines. [Figure 34–4](#) shows the necessary pipelines and account router data pool.

Figure 34–4 Account Router Pipeline Manager



For information on how to configure the account-router Pipeline Manager, see ["Configuring Your Account-Router Pipeline Manager"](#).

About Suspending Call Records

The account-router Pipeline Manager initially routes and suspends call records in the *routing pipeline*.

After AMM notifies the account-router Pipeline Manager that a job is being migrated, the routing pipeline performs the following:

1. (Optional) The FCT_CallAssembly module assembles EDRs that were split into multiple records.

Important: Any call assembling must occur in the account router instance of the Pipeline Manager and not in other instances.

For more information, see "FCT_CallAssembling" in *BRM Configuring Pipeline Rating and Discounting*.

2. (Optional) The FCT_DuplicateCheck module checks whether EDRs have been previously rated by the Pipeline Manager.

Important: Any checking for duplicate EDRs must occur in the account router instance of the Pipeline Manager and not in other instances.

See "FCT_DuplicateCheck" in *BRM Configuring Pipeline Rating and Discounting*.

3. The FCT_PreSuspense module adds suspense-related data to the EDR.
4. The FCT_AccountRouter module, set to **Router** mode:
 - Flags the EDR for the target Pipeline Manager.
 - Determines whether an EDR is for an account undergoing migration. If it is, FCT_AccountRouter flags the EDR for suspension. See "FCT_Account" in *BRM Configuring Pipeline Rating and Discounting*.
5. The FCT_Reject module routes EDRs with a specified error status, such as warning or critical, to the suspense output stream.
6. The FCT_Suspense module determines whether an EDR is flagged for suspension. If it is, FCT_Suspense places the EDR in a separate suspense output stream, where it is eventually loaded into the BRM database by the Suspense Event (SE) Loader.

For more information about recycling suspended EDRs in BRM, see "About the EDR Recycling Features" in *BRM Configuring Pipeline Rating and Discounting*.

Note: You can use either standard recycling or Suspense Manager with AMM.

About Reprocessing Suspended Call Records

After AMM successfully migrates a job, it calls the PCM_OP_SEARCH_RECYCLE opcode to recycle previously suspended EDRs through the pipeline. Then, the account-router Pipeline Manager recycles suspended EDRs through the *pre-recycling pipeline* and the *resuspending pipeline*.

The account-router Pipeline Manager recycles EDRs as follows:

1. In the pre-recycling pipeline, the FCT_AccountRouter module, set to **Recycle** mode, determines whether an EDR is targeted for an account that is being migrated by a new job.

- If the account *is* being migrated by a new job, FCT_AccountRouter flags the EDR for suspension and routes the EDR to a separate suspense output stream, where it is processed by the resuspending pipeline.
 - If the account *is not* being migrated, FCT_AccountRouter flags the EDR for the appropriate instance of the Pipeline Manager. The EDR is then rated by the target Pipeline Manager.
2. The resuspending pipeline automatically routes EDRs to a separate suspense output stream, which is eventually loaded into the BRM database by Suspense Event (SE) Loader.
 - a. The FCT_PreSuspense module adds suspense-related data to the EDR.
 - b. The ISC_Migration iScript automatically flags the EDR for suspension.
 - c. The FCT_Reject module routes EDRs with a specified error status to the suspense output stream.
 - d. The FCT_Suspense module routes the EDR to a suspense output stream, which is eventually loaded into the BRM database by SE Loader.

For more information about Suspense Manager, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Your System to Migrate Accounts When the Pipeline Manager Is Running

You configure your BRM system to migrate accounts when your pipelines are online by:

1. [Configuring Your Account-Router Pipeline Manager](#)
2. [Configuring BRM to Handle Suspended EDRs](#)
3. [Configuring AMM to Send Business Events to Your Pipelines](#)
4. [Configuring Your Pipelines to Send Acknowledgments to AMM](#)

Configuring Your Account-Router Pipeline Manager

To configure your account router instance of the Pipeline Manager, perform the following:

- [Configuring Your Routing Pipeline](#)
- [Configuring Your Pre-recycling Pipeline](#)
- [Configuring Your Resuspending Pipeline](#)
- [Configuring the Data Pool](#)

Configuring Your Routing Pipeline

You configure your routing pipeline to route and suspend EDRs by using the following pipeline modules:

- **FCT_PreSuspense.** To make suspense fields queryable in Suspense Management Center, set the following FCT_PreSuspense registry entries:
 - Use the **Active** entry to enable this module.
 - Use the **QueryableFields** entry to specify the tables and fields that you can perform queries on in Suspense Management Center.

See "FCT_PreSuspense" in *BRM Configuring Pipeline Rating and Discounting*.

- **FCT_AccountRouter** set to **Router** mode. To flag EDRs for suspension and for the appropriate Pipeline Manager, set the following FCT_AccountRouter registry entries:
 - Use the **Active** entry to enable this module.
 - Use the **Mode** entry to specify **Router** mode.
 - Use the **Streams** entry to map EDRs to the appropriate output stream.

See "FCT_AccountRouter" in *BRM Configuring Pipeline Rating and Discounting*.

- **FCT_Reject**. To route EDRs with a specified error status to the suspense output stream:
 - Use the **Active** entry to enable this module.
 - Set the **UseRejectStream** entry to **True**. This sends EDRs to the reject stream.
 - Use the **MinErrorSeverity** entry to reject EDRs that have the specified error severity.
 - Use the **StreamMap** entry to map errors to specific output streams.

Important: You must also configure an instance of the Out_Reject module for rejected EDRs. All rejected EDRs must be set to the suspense output stream. See "OUT_Reject" in *BRM Configuring Pipeline Rating and Discounting*.

See "FCT_Reject" in *BRM Configuring Pipeline Rating and Discounting*.

- **FCT_Suspense**. To send EDRs to the suspense output stream, set the following FCT_Suspense registry entries:
 - Use the **Active** entry to enable this module.
 - Use the **SuspenseCreateStream** entry to specify the output stream for suspended EDRs.
 - Use the **SuspenseUpdateStream** entry to specify the output stream for recycled EDRs.
 - Use the **DataConnection** entry to specify how to connect to the BRM database.

See "FCT_Suspense" in *BRM Configuring Pipeline Rating and Discounting*.

If you want your pipelines to assemble EDRs or check for duplicate EDRs, you must also use the FCT_CallAssembly and FCT_DuplicateCheck modules in the routing pipeline. See "FCT_CallAssembling" and "FCT_DuplicateCheck" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Your Pre-recycling Pipeline

You configure your pre-recycling pipeline to recycle or suspend EDRs by using the FCT_AccountRouter module set to **Recycle** mode. Make sure you also set the following FCT_AccountRouter registry entries:

- Use the **Active** entry to enable this module.
- Use the **Mode** entry to specify **Recycle** mode.
- Use the **Streams** entry to map EDRs to the appropriate output stream.

See "FCT_AccountRouter" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Your Resuspending Pipeline

You configure your resuspending pipeline to automatically suspend all EDRs by using the following pipeline modules:

- **FCT_PreSuspend.** To make suspense fields queryable in Suspense Management Center, set the following FCT_PreSuspend registry entries:
 - Use the **Active** entry to enable this module.
 - Use the **QueryableFields** entry to specify the tables and fields that you can perform queries on in Suspense Management Center.

See "FCT_PreSuspend" in *BRM Configuring Pipeline Rating and Discounting*.

- **ISC_Migration.** To automatically flag all EDRs for suspension, set the following ISC_Migration registry entries:

- Use the **Active** entry to enable this module.
- Use the **Filename** entry to specify the path to the ISC_Migration file.

See "ISC_Migration" in *BRM Configuring Pipeline Rating and Discounting*.

- **FCT_Reject.** To route EDRs with a specified error status to the suspense output stream:
 - Use the **Active** entry to enable this module.
 - Set the **UseRejectStream** entry to **True**. This sends EDRs to the reject stream.
 - Use the **MinErrorSeverity** entry to reject EDRs that have the specified error severity.
 - Use the **StreamMap** entry to map errors to specific output streams.

Important: You must also configure an instance of the Out_Reject module for rejected EDRs. All rejected EDRs must be set to the suspense output stream. See "OUT_Reject" in *BRM Configuring Pipeline Rating and Discounting*.

See "FCT_Reject" in *BRM Configuring Pipeline Rating and Discounting*.

- **FCT_Suspend.** To send EDRs to the suspense output stream, set the following FCT_Suspend registry entries:
 - Use the **Active** entry to enable this module.
 - Use the **SuspenseCreateStream** entry to specify the output stream for suspended EDRs.
 - Use the **SuspenseUpdateStream** entry to specify the output stream for recycled EDRs.
 - Use the **DataConnection** entry to specify how to connect to the BRM database.

See "FCT_Suspend" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring the Data Pool

You configure the account-router Pipeline Manager data pool to pass account migration data to your pipelines by using the following pipeline data modules:

- **DAT_AccountBatch** stores AMM business events. In addition to setting the standard connection registry entries:
 - Set the **UseAsRouter** entry to **True**. This is required.
 - (Optional) Use the **PrintAMTData** entry to specify whether to print AMM data to a log file. You can use this data for troubleshooting.
 - (Optional) Use the **PrintAMTJobData** entry to specify whether to print data about one migration job to a log file. You can use this data for troubleshooting.

See "DAT_AccountBatch" in *BRM Configuring Pipeline Rating and Discounting*.

- **DAT_BalanceBatch**. To provide accurate account balances during migration, set the following DAT_BalanceBatch registry entries:
 - Use the **IntegrateConnection** entry to specify how to connect to the pipeline database. This entry points to the **Login** registry section.
 - Use the **InfranetConnection** entry to specify how to connect to the BRM database. This entry points to the **LoginInfranet** registry section.
 - Use the **ListenerDataModule** entry to specify how to connect to the DAT_Listener module. This entry points to the **Listener** registry section.

See "DAT_BalanceBatch" in *BRM Configuring Pipeline Rating and Discounting*.

- **DAT_Listener**. To retrieve business events from BRM and send acknowledgment events directly to the acknowledgment queue, set the following DAT_Listener registry entries:
 - Use the **InfranetConnection** entry to specify how to connect to the database schema that contains your queues. This entry points to the **LoginInfranet** registry section.
 - Use the **AckQueueNameAMM** entry to specify the name of the acknowledgment queue.
 - Use the **QueueName** entry to specify the name of the Account Synchronization queue that stores the AMM business events.

See "DAT_Listener" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring BRM to Handle Suspended EDRs

BRM offers both the default *standard recycling* feature and the optional *Suspense Manager* feature to recycle EDRs. For a comparison of the two, see "About the EDR Recycling Features" in *BRM Configuring Pipeline Rating and Discounting*.

AMM works with both standard recycling and Suspense Manager.

Both standard recycling and Suspense Manager enable you to do the following:

- Load suspended EDRs into the BRM database.
- View, edit, write off, or recycle suspended EDRs.
- Retrieve suspended EDRs from the BRM database.

You must configure your pipeline before you can migrate accounts with the pipeline running. For information on how to configure your pipeline, see "Configuring Standard Recycling" in *BRM Setting Up Pricing and Rating*. If you have purchased Suspense Manager, see the Suspense Manager documentation for further configuration instructions.

Configuring AMM to Send Business Events to Your Pipelines

To configure AMM to send business events to your pipelines, perform the following:

1. [Connecting AMM to the Primary CM](#)
2. [Configuring Account Synchronization](#)
3. [Configuring Your Pipelines to Dequeue AMM Business Events](#)

Connecting AMM to the Primary CM

You connect AMM to the primary Connection Manager (CM) so that AMM can send business events to the Account Synchronization architecture, where they are eventually routed to your pipelines.

You connect AMM to the primary CM by using the **infranet.connection** and **infranet.login.type** parameters in the AMM **Infranet.properties** file. See "[Connecting AMM to Your Database Schemas](#)".

Configuring Account Synchronization

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

1. Configuring Account Synchronization to send BRM events to your pipelines. To do this, follow the instructions in "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*.
2. Creating queues for sending AMM business events to your pipelines. You must create a queue for each Pipeline Manager instance in your system. See "[Configuring Your Account Synchronization Queues for AMM Business Events](#)".
3. Mapping the AMM business events to your Oracle database queues. See "[Mapping AMM business events to your queues](#)".

Configuring Your Account Synchronization Queues for AMM Business Events

The Account Synchronization framework uses a set of Oracle database queues to send both BRM events and AMM business events to your pipelines.

Each instance of the Pipeline Manager must have its own database queue. For example, if your system contains three BRM database schemas, Account Synchronization requires a total of four queues. That is, one for each of the following instances:

- Account-router Pipeline Manager
- Pipeline Manager for BRM database schema 1
- Pipeline Manager for BRM database schema 2
- Pipeline Manager for BRM database schema 3

Important: You must also create a separate queue for any external applications that require notifications about account migration.

If your system does not already contain a queue for each instance of the Pipeline Manager, you can create additional queues by using the Account Synchronization **pin_ifw_sync_oracle** utility. For more information about creating queues, see "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*.

Mapping AMM business events to your queues

You map which types of events Account Synchronization sends to your queues by using the **ifw_sync_queuenames** file (*BRM_home/sys/dm_ifw_sync/ifw_sync_queuenames*). This file lists all queues in your system and the events to route to each one.

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

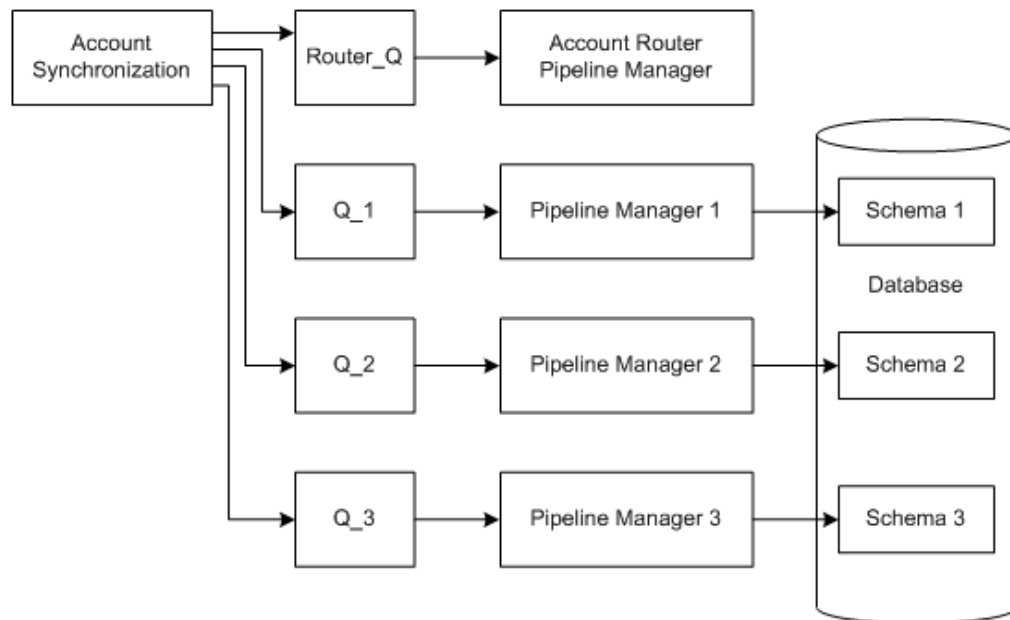
```
QueueName@DatabaseLink
{
    Criteria
}
```

Where:

- *QueueName* specifies the name of the queue.
- *DatabaseLink* specifies the database link for connecting to queues on other database schemas. Provide a link only for queues that reside on a separate schema from which the Account Synchronization DM connects.
- *Criteria* specifies which events to send to the queue. You can configure the Account Synchronization DM to send all business events, only events from a specific database schema, or only specific event types.

For example, assume your BRM system contains three database schemas and four queues as shown in [Figure 34-5](#):

Figure 34-5 Sample Pipeline Account Synchronization Architecture



In this system, you must configure the **ifw_sync_queuenames** file so that the Account Synchronization architecture does the following:

- Sends **HoldCDRProcessing**, **ResumeCDRProcessing**, and **MigrateAcct** events to the queue for the account-router Pipeline Manager
- Sends **MigrateSource** and **MigrateDestination** events to each queue connected to a BRM database schema

For more information about AMM business events, see ["About Notifying the Pipelines about Account Migration"](#).

In this example, the `ifw_sync_queuenames` file requires the following additional entries:

```
ROUTER_Q
{
    HoldCDRProcessing
    ResumeCDRProcessing
    MigrateAcct
}

Q_1 # local database schema queue
{
    MigrateSource
    MigrateDestination
}

Q_2@database_link_1 # remote database schema
{
    MigrateSource
    MigrateDestination
}

Q_3@database_link_2 # remote database schema
{
    MigrateSource
    MigrateDestination
}
```

Configuring Your Pipelines to Dequeue AMM Business Events

You must configure *each instance* of the Pipeline Manager to retrieve AMM business events from the Account Synchronization queue. To do this, connect the `DAT_Listener` modules to your Account Synchronization queues by setting the following registry entries:

- Use the **InfranetConnection** entry to specify how to connect to the database schema that contains the Account Synchronization queue. This entry points to the **LoginInfranet** registry section.
- Use the **QueueName** entry to specify the name of the Account Synchronization queue that holds the AMM business events.

See "DAT_Listener" in *BRM Configuring Pipeline Rating and Discounting*.

Configuring Your Pipelines to Send Acknowledgments to AMM

You configure your pipelines to send acknowledgment events to a centralized queue, where they are retrieved by the AMM Controller, by performing the following:

1. [Creating the Acknowledgment Queue](#)
2. [Connecting AMM Directly to Your Acknowledgment Queue](#)
3. [Configuring Your Pipelines to Send Acknowledgment Events](#)

Creating the Acknowledgment Queue

You create a centralized acknowledgment queue for sending events from your pipelines to the AMM Controller.

You create the acknowledgment queue by using the **pin_ifw_sync_oracle** utility. Enter the following commands at a UNIX prompt:

```
% su - pin
% cd BRM_home/apps/pin_ifw_sync
% pin_ifw_sync_oracle.pl create [-l username/password@DatabaseAlias] [-q queue_name -t queue_table]
```

The utility creates a database queue named IFW_SYNC_QUEUE and a queue table named IFW_SYNC on the specified schema. To use nondefault names, use the **-q** and **-t** options to specify names for the queue and queue table.

Important: In multischema systems, all queues and queue tables must use unique names. You must also make sure the acknowledgment queue is accessible by the **pin** user.

For more information, see the following documents:

- "pin_ifw_sync_oracle" in *BRM Installation Guide*
- "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*

Connecting AMM Directly to Your Acknowledgment Queue

You connect the AMM Controller to the acknowledgment queue so that it can retrieve acknowledgment events.

You connect the AMM Controller to the acknowledgment queue by using the **controller_N_amt_queue_name** and **controller_N_amt_queue_owner_name** entries in the AMM **Infranet.properties** file. See ["Connecting AMM to Your Database Schemas"](#).

Configuring Your Pipelines to Send Acknowledgment Events

You configure your pipelines to send acknowledgment events to AMM by configuring the DAT_Listener module in each Pipeline Manager.

Configure the DAT_Listener registry entries in *each* instance of the Pipeline Manager to specify the following:

- Use the **InfranetConnection** entry to specify how to connect to the database schema that contains your acknowledgment queue.
- Use the **AckQueueNameAMM** entry to specify the name of the acknowledgment queue. This is the queue you created in ["Creating the Acknowledgment Queue"](#).

See "DAT_Listener" in *BRM Installation Guide*.

Using Account Migration Manager

This chapter describes how to use the Oracle Communications Billing and Revenue Management (BRM) Account Migration Manager (AMM) software to migrate accounts from a source database schema to a destination database schema in the same database. It also describes how to perform such account migration when you use Oracle IMDB Cache in your environment.

Overview of Account Migration Tasks

Migrating accounts includes the following general tasks. Although you can perform some tasks at any time, the following order is recommended:

1. If you use Oracle IMDB Cache in your system, verify that you have configured your environment appropriately.
See ["Verifying Your System Configuration When Oracle IMDB Cache Is Used"](#) for more information.
2. Create the account search configuration file.
See ["Creating the Account Search Configuration File"](#) for more information.
3. Submit the account migration job.
See ["Submitting the Account Search File"](#) for more information.
4. For account group migration, run a **group_details** report to verify that each account group includes all account members.
See ["Checking Account Group Details"](#) for more information.

Caution: You must verify that the job includes all accounts in the account group. Any missing accounts will be stored in a separate database schema from the account group, which severs the account's relationship with the group.

5. Enable the account migration job.
See ["Enabling Migration Jobs in the Queue"](#) for more information.
6. Start the AMM Controller.
See ["Starting the AMM Controller"](#) and ["Monitoring the AMM Controller"](#) for more information.
7. Monitor the job's progress.
See ["Monitoring Account Migration"](#) for more information.

8. Fix account migration failures, when necessary.
See ["Handling Account Migration Failures"](#) for more information.
 9. Purge the migrated accounts from the source database schema.
See ["Purging Migrated Objects from the Source Database Schema"](#) for more information.
 10. Stop the AMM Controller.
See ["Stopping the AMM Controller"](#) for more information.
 11. If you use Oracle IMDB Cache in your system, load the data for the migrated accounts into the destination cache grid.
See ["Loading Destination Oracle IMDB Cache with Data from BRM Database"](#) for more information.
- See ["Deleting Jobs from the Source Database Schema"](#) for information on deleting jobs from the source database schema.

Verifying Your System Configuration When Oracle IMDB Cache Is Used

Verify the following before you proceed with the account migration:

1. The TimesTen JDBC Driver Jar files are available for every instance of BRM in your environment. See ["Configuring the TimesTen JDBC Driver Jar file for BRM"](#).
2. Verify that the **Infranet.properties** file is configured appropriately. See ["Configuring the AMM Infranet.properties File"](#).
3. Verify that the **load_pin_uniqueness** utility has been configured correctly. See ["Configuring the load_pin_uniqueness Utility for Oracle IMDB Cache"](#).

Creating the Account Search Configuration File

You use the account search configuration file to specify the source and destination database schemas, the search criteria, the maximum number of accounts in a job, and the number of accounts in each batch.

AMM can search for accounts that meet five default criteria:

- Account creation date
- Account status
- Billing day of month
- Product name
- POID

If you would like to migrate accounts that meet some other custom criteria, see ["Creating Custom Account Search Criteria"](#).

To create an account search configuration file:

1. Copy the sample account search configuration file (*BRM_home/apps/amt/account_search.cfg*) and save it with another name. Use this file, which contains all of the configuration entries, as a template.
2. Edit the entries listed in [Table 35–1](#) in the file.

Note: Only the source database schema, destination database schema, batch size, and one other entry is required. If you do not want to use an entry, leave it blank.

Table 35–1 *account_search.cfg Parameters*

Parameter	Description	Required
src_database	Specifies the source database schema, which is the schema from which you are migrating accounts. For example, enter 0.0.0.1 . This value must match one of the database numbers specified in the Infranet.properties file.	Yes
dest_database	Specifies the destination database schema, which is the schema to which you are migrating accounts. For example, enter 0.0.0.2 . This value must match one of the database numbers specified in the Infranet.properties file.	Yes
start_creation_date	Use this parameter to migrate accounts that were created in a specific date range. AMM migrates accounts created between midnight (00:00:00) on the start date and 23:59:59 on the end date. For example, to migrate accounts created after midnight on August 1, 2004, enter 08/01/2004 . Important: If you set this parameter, you must also set the end_creation_date parameter.	No
end_creation_date	Use this parameter to migrate accounts that were created in a specific date range. AMM migrates accounts created between midnight (00:00:00) on the start date and 23:59:59 on the end date. For example, to migrate accounts created on or before 11:59:59 p.m. on August 10, 2004, enter 08/10/2004 . Important: If you set this parameter, you must also set the start_creation_date parameter.	No
migration_mode	Specifies whether to migrate account groups. When AMM finds an account that belongs to a hierarchy, sponsorship, or resource sharing group, AMM migrates all accounts related to that account. <ul style="list-style-type: none">■ IncludeAccountGroup specifies to migrate accounts groups.■ ExcludeAccountGroup specifies to exclude account groups from migrations. The default is ExcludeAccountGroup . Important: If you set this parameter, you must also set the max_group_size parameter.	No
max_group_size	Specifies the maximum size of an account group that AMM can migrate. If an account group exceeds the maximum number of accounts, AMM excludes the account group from the job. The default is 100 .	No
product_name	Migrates accounts that purchased the specified product. For example, Product 1b - Email Account .	No
account_status	Migrates accounts based on the specified account status. <ul style="list-style-type: none">■ Active specifies to migrate active accounts only.■ Inactive specifies to migrate inactive accounts only.■ Closed specifies to migrate closed accounts only.	No
bill_day_of_month	Migrates accounts that have the specified billing day of month. You can specify any number from 1 through 31. For example, enter 4 to migrate all accounts that are billed on the 4th of the month.	No

Table 35–1 (Cont.) account_search.cfg Parameters

Parameter	Description	Required
max_accounts	Specifies the maximum number of accounts to move in a job.	No
batch_size	<p>Specifies the number of accounts in each batch. You can specify any amount from 1 through 1,000. However, for optimal performance, set this to an integer between 50 and 100.</p> <p>Important:</p> <ul style="list-style-type: none"> Using a batch size of more than 50 accounts does not improve performance. If you set this to a number greater than 100, you must increase the size of your Oracle rollback segments. For more information, contact your Oracle BRM representative. 	Yes
poid_list	Migrates accounts based on the POID. Use comma separators, for example, 22860, 22861, 22862 . Limit the number of accounts to 1,000 or less.	No

3. Save the file.

Sample Account Search Configuration File

The following sample account search configuration file specifies to:

- Migrate accounts from database schema **0.0.0.1** to database schema **0.0.0.2**.
- Migrate in batches of 50 accounts.
- Migrate only nonmember accounts.
- Migrate accounts that meet the following criteria:
 - Created between January 1, 2004 and June 31, 2004
 - Have an active account status
 - Purchased the **Product 1b - Email Account** product

```
src_database=0.0.0.1
dest_database=0.0.0.2
start_creation_date=01/01/2004
end_creation_date=06/31/2004
migration_mode=ExcludeAccountGroup
max_group_size=
product_name=Product 1b - Email Account
account_status=Active
bill_day_of_month=
max_accounts=
batch_size=50
poid_list=
```

Submitting the Account Search File

When you submit an account search file, the **pin_amt** utility searches the source database schema and populates the job management tables on the primary, source, and destination database schemas with a list of accounts meeting the specified criteria.

1. Submit your account search information to the **pin_amt** utility:

```
% pin_amt -s AccountSearchFile
submitted job
job_id=30
```

The **pin_amt** utility notifies you if it successfully submitted the file and gives you the job ID number.

2. Write down the job ID number, because you will need it later.

Enabling Migration Jobs in the Queue

The AMM Controller can only begin processing an account migration job after it's enabled in the queue.

To enable a job in the queue, enter this command:

```
% pin_amt -e JobID
enabled job
```

Starting the AMM Controller

After it is started, the AMM Controller runs as a server process, continuously checking for jobs to process in the queue.

To start the AMM Controller, enter this command:

```
% pin_amt -c start [-a ControllerID]
controller is started
controller_id=1
```

Note: If your system contains multiple AMM Controllers, use the **-a** option to specify which AMM Controller to start. By default, **pin_amt** starts Controller 1.

The **pin_amt** utility notifies you if the AMM Controller started successfully and which AMM Controller is active.

Monitoring the AMM Controller

You can monitor the AMM Controller's status at any time by using the **pin_amt** utility or checking the AMM Controller log file.

Checking the AMM Controller Status

To check whether the AMM Controller is up and running, enter this command:

```
% pin_amt -c status [-a ControllerID]
controller status is up
```

Note: If your system contains multiple AMM Controllers, use the **-a** option to specify which AMM Controller to check.

The **pin_amt** utility notifies you that the AMM Controller is up or down. If the AMM Controller is down or cannot be started, check the AMM Controller log file for more information.

Checking the AMM Controller Log File

The AMM Controller log file contains a detailed list of all transactions executed by the AMM Controller. This log is created in the directory specified in the **controller_N_log_directory** entry of the **Infranet.properties** file. You can open the log file by using a text editor.

Monitoring the AMM Controller in Real Time

You can use the **pin_amt** utility to see what the AMM Controller is doing in real time.

To monitor the AMM Controller in real time, enter this command:

```
% pin_amt -c log [-a ControllerID]
```

Note: If your system contains multiple AMM Controllers, use the **-a** option to specify which AMM Controller to check.

A separate Xterm window opens. For best viewing, set the Xterm width to 120. If an Xterm window fails to open, make sure your **DISPLAY** environment variable is set correctly.

Monitoring Account Migration

You can monitor the status of jobs in the queue by running three special AMM reports: **list_jobs**, **job_details**, and **group_details**.

Monitoring Job Status

The **list_jobs** report provides the status of each job in the queue, including the number of batches that failed to migrate.

To run the **list_jobs** report, enter this command:

```
% pin_amt -r list_jobs
```

Sample output from a **list_jobs** report:

```
Tue Mar 12
```

page 1

```

                                AMT jobs
                                -----
                                Total   Failed   Succ.
                                Account Account Account
                                -----
creation
Job Name  User Name  Job ID  batches  batches  batches  Job
Status   time[sec]  time           Accounts
-----
test.cfg  pin          1       6       0       6  FINISHED      40  02/02/2002
13:39    100
srch.cfg  pin          2       3       0       3  FINISHED      25  02/15/2002
12:00    50
mar1.cfg  pin          3       8       1       7  FINISHED     205  03/01/2002
18:42    400

```

If any batches failed, you can see greater detail on why the batch failed by running the **job_details** report.

Checking Job Details

The **job_details** report provides detailed information about a job's status, including why a batch failed.

To run the **job_details** report, enter this command:

```
% pin_amt -r job_details
enter job id:
```

Sample output from a **job_details** report:

Tue Mar
12

page 1

AMT job details

Job ID	Account batches	Status	Error	Processing start	Batch processing	
Message		date		time[sec]	Accounts	
-						
0	3	1	FINISHED	03/01/2002 18:42	25	5
distributed		2	FAILED			
		03/01/2002 18:42	ORA-02055: update operation failed; rollback required ORA-02049: timeout: distributed transaction waiting for lock ORA-06512: at "PIN.AMT_MV", line 454 ORA-06512: at line 1	5	0	
0		3	FINISHED	03/01/2002 18:42	25	5
0		4	FINISHED	03/01/2002 18:43	25	5
0		5	FINISHED	03/01/2002 18:43	25	5
0		6	FINISHED	03/01/2002 18:44	25	5
0		7	FINISHED	03/01/2002 18:44	25	5
0		8	FINISHED	03/01/2002 18:45	25	5

The report lists any error messages from the Oracle database. For information, see the Oracle documentation.

Checking Account Group Details

The **group_details** report lists the accounts in each account group and provides information about each group's migration status. You use this information to verify that all account members are included in a group.

Caution: All accounts in a hierarchy, sponsorship, or resource sharing group must reside in the same database schema. Any accounts separated from a parent account will no longer be associated with the account group.

To run the **group_details** report, enter this command:

```
% pin_amt -r group_details
enter job id:
enter group id:
```

Sample output from a **group_details** report:

```
Tue Mar 12
AMT group details
page 1
```

Job ID	Account batches	Batch Status	Group ID	Group Status
3	1	FINISHED	1	NOT_PROCESSED

```
Tue Mar 12
AMT group member details
page 1
```

Account batches	Accounts ID	Accounts DB
1	17009	2
1	17289	2
1	16489	2
1	17313	2
1	16465	2
1	17066	2

Handling Account Migration Failures

An account batch may fail for several reasons. The most common reasons are as follows:

- An application is accessing or modifying the data you are attempting to migrate.
- The database is down.

Finding Debugging Information

For information on why a batch failed, you can run a **job_details** report or check any of the following files, which are located in the directories you specified in the **Infranet.properties** file.

- AMM installation log file (**pin_amt_install.log**)
- AMM Controller log file (**controller_N_YYYYMMDDhhmm.log**)
- **pin_amt** log file (**pin_amt.log**)
- AMM configuration file (**Infranet.properties**)
- Account search configuration file (**account_search.cfg**)
- AMM Mover log files (**amt_migrate_JobID_BatchNumber.log**)
- AMM delete log file (**amt_delete_JobID_BatchNumber.log**)

If you need assistance in resolving migration failures, send these files along with any additional information about the problem to your Oracle BRM representative.

Reprocessing Failed Batches

To reprocess a batch that failed:

1. Fix the problem.
2. Change the status of the batch from FAILED to NOT PROCESSED:

```
% pin_amt -b JobID:BatchNumber
```

3. Enable the job in the queue again:

```
% pin_amt -e JobID
```

The AMM Controller processes all batches that have a NOT PROCESSED status and ignores batches with a FINISHED status.

Purging Migrated Objects from the Source Database Schema

After you successfully migrate your accounts, you can improve your overall system performance by purging the migrated (invalid) objects from your source database schema. Also, because the purging process uses only one thread, purges accounts sequentially, and does not affect data used by BRM, you can purge accounts at any time.

To purge successfully migrated objects from the source database schema, enter this command:

```
pin_amt -p SourceDatabaseSchema
```

Deleting Jobs from the Source Database Schema

You can use the delete option to:

- Remove both failed and successfully migrated jobs from your database schemas
- Free up disk space

The delete option performs the following actions listed in [Table 35–2](#):

Table 35–2 Delete Job Actions

Job Type	Action
Failed jobs	Deletes the job from the AMM job management tables.
Successfully migrated jobs	<ul style="list-style-type: none"> ■ Deletes the job from the AMM job management tables. ■ Deletes account-related data from the source database schema.

To delete a job, run the **pin_amt** script with the delete option:

```
pin_amt -d JobID
```

Stopping the AMM Controller

You can stop the AMM Controller at any time. If you stop the AMM Controller while it is processing a batch, it finishes the batch before stopping.

To stop the AMM Controller, enter this command:

```
% pin_amt -c stop
```

```
controller is stopped
controller_id=1
```

Pausing and Resuming Account Migration

If a job contains a large number of accounts, but you only have a limited amount of time in which to migrate accounts, you can migrate the job in stages. The AMM software enables you to start an account migration job and then pause it when your window of opportunity is over. When you reach the next window of opportunity, you can resume the job where it left off.

Note: An AMT Controller must completely finish migrating one job before it can start migrating another job. Therefore, if you pause one job and then enable a second job, the AMT Controller cannot begin processing the second job until the first job is finished.

To pause an account migration job, enter this command:

```
% pin_amt -c pause
paused controller
controller_id=1
```

To resume an account migration job, enter this command:

```
% pin_amt -c continue
continued controller
controller_id=1
```

Loading Destination Oracle IMDB Cache with Data from BRM Database

Load the destination Oracle IMDB Cache with the migrated accounts from the destination BRM database schema only after the **pin_amt** utility successfully migrates the accounts into the destination BRM database schema.

Note: For each migrated account, the AMM Mover updates the account POIDs in the uniqueness table to reflect the account's new location.

After you complete this step, the critical data for the migrated accounts will be in the appropriate Oracle IMDB Cache (data store) ready for use.

To load the account data from the destination BRM database schema into the appropriate Oracle TimesTen Cache (data store) in the destination Oracle IMDB Cache grid, you must run the appropriate **tt_load_Logical_Partition.sql** script. The **tt_load_Logical_Partition.sql** scripts were generated on the destination Oracle IMDB Cache during the installation of Oracle TimesTen.

For example, the destination Oracle TimesTen Cache (data store) for a migration task is identified as **tt_0.1.0.1** (associated with the logical partition identified as **0.1.0.1**). After you migrate accounts to the BRM database schema associated with *this* destination Oracle IMDB Cache, you must run **tt_load_0.1.0.1.sql** on the **0.1.0.1** logical partition of this cache grid.

If some accounts were migrated to one Oracle IMDB Cache and other accounts were migrated to a different Oracle IMDB Cache, then you must load the data for the

migrated accounts from the BRM database schema into the *appropriate* cache. To do so, you the run **tt_load_Logical_Partition.sql** script for every (destination) data store which received the migrated accounts.

Note: If necessary, run the **pin_tt_schema_gen** utility against the destination Oracle IMDB Cache grid to generate the required **tt_load_Logical_Partition.sql** scripts for the logical partitions.

Complete the steps in the following procedure only after the account migration is successfully completed. To load subscriber data from the BRM database schema into the destination Oracle IMDB Cache:

1. Go to the **bin** directory where the active data store resides:

```
cd IMDB_home/bin
```

where *IMDB_home* is the directory in which Oracle IMDB Cache is installed.

2. Connect to the destination Oracle IMDB Cache using **ttisql**:

```
ttisql Data_Store_Name
```

where *Data_Store_Name* is the name of the data store, such as **tt_0.1.0.1**

3. Enter the following command to run the required **tt_load.sql** script against the destination Oracle IMDB Cache data store:

```
run BRM_home/bin/tt_load_Logical_Partition.sql;
```

where *Logical_Partition* is the database number of the logical partition associated with the data store, such as **0.1.0.1**

See "[Loading Subscriber Data into the Oracle IMDB Cache Data Stores](#)" for information.

Automating Account Migration

You can use an external scheduler, such as **cron**, to automate account migration during your maintenance window. If you must migrate a large number of accounts, you can set up **cron** to stop and restart account migration at specific times.

For example, scheduling account migration for every Sunday from 2:00 a.m. to 4:00 a.m. requires these tasks:

1. Stop the AMM Controller.
2. Create your account search configuration files.
3. Submit your jobs.
4. Enable your jobs in the queue.
5. Configure one **cron** job to start the AMM Controller every Sunday at 2:00 a.m. and check for errors. See "[AMM Return Codes and Messages](#)".
6. Configure a second **cron** job to stop the AMM Controller every Sunday at 4:00 a.m. and check for errors.

Migrating Accounts within an Oracle IMDB Cache Grid

This chapter describes how to migrate Oracle Communications Billing and Revenue Management (BRM) accounts located within the same Oracle IMDB Cache grid.

Before attempting to migrate such account data, you should be familiar with the following topics:

- Account Migration. See ["Understanding Account Migration"](#).
- Oracle IMDB Cache Manager. See ["Managing IMDB Cache-Enabled Systems"](#).

About Account Migration within an Oracle IMDB Cache Grid

Migration of account data within an Oracle IMDB Cache grid consists of migrating the data associated with selected accounts from a logical partition in the cache grid to a destination logical partition located in the *same* cache grid.

When to Migrate Accounts within an Oracle IMDB Cache Grid

You migrate accounts between two logical partitions in one Oracle IMDB Cache grid when accounts drastically increase (or decrease) in one logical partition in the cache grid or when you add a logical partition to the cache grid. See ["Using the Oracle IMDB Cache Grid to Partition Data"](#) for more information.

When you add a logical partition to an Oracle IMDB Cache grid, you migrate accounts from each of the existing logical partitions to the newly-created logical partition to reset the load balance in that cache grid.

For example, if your system uses logical partitioning and you have two logical partitions in your database schema, the first logical partition is identified as **0.0.0.1** and the second logical partition as **0.1.0.1**. The data stores (that is, the Oracle IMDB Caches) associated with these logical partitions are identified as **tt_0.0.0.1** and **tt_0.1.0.1** respectively.

Continuing with this example, suppose that you maintain 100 accounts in that Oracle IMDB Cache grid by storing 50 accounts in the data stores associated with each of its two logical partitions. A promotion effort results in the addition of 50 new accounts to that cache grid. To better manage your customer accounts, you add a logical partition (identified as **0.2.0.1**) to that cache grid. You then select 25 accounts from each of the existing partitions to be migrated to populate the new logical partition you added. You migrate the selected accounts by performing two migration operations, one from each of the existing Oracle IMDB Caches (logical partitions) to the new cache (the logical partition identified as **0.2.0.1**) in that cache grid. At the end of the two-step operation,

the three data stores associated with the logical partitions in the cache grid carry an even load (with each data maintaining the data for 50 accounts).

How BRM Migrates Account Data within an Oracle IMDB Cache Grid

BRM provides the **pin_amt_tt** utility for the migration of accounts between logical partitions in the same IMDB Cache grid.

About the **pin_amt_tt** Utility

The **pin_amt_tt** utility is a standalone utility and is installed as part of the Account Migration Manager installation. It is used to move accounts between the data stores associated with one Oracle IMDB Cache grid only.

The **pin_amt_tt** utility migrates the accounts using the values in the **pin.conf** configuration file that you provide for that migration task. This configuration file informs the utility about the accounts to migrate and the access information for the source and destination locations within that cache grid.

1. At the start of the migration, the **pin_amt_tt** utility fetches the list of account POIDs it must migrate. This list of accounts is based on the selection criteria you provided in the utility's **pin.conf** file.
2. It locks the associated cached information on the selected accounts to prevent all authentication, authorization, and accounting (AAA) activity on those accounts. If any non-AAA operations are currently being performed on those accounts, BRM waits for those processes to finish.
3. The utility then makes the entries in the uniqueness table (in the BRM database) invalid for these accounts so as to prevent access to the data in them through any uniqueness reference.
4. It unloads the accounts from the source Oracle IMDB Cache, ensures that the BRM database contains the current data, and loads them into the destination Oracle IMDB Cache.
5. The utility updates the entries in the uniqueness table (in the BRM database) to point each migrated account to its appropriate destination cache location.

For example, for an account **2286** located in the logical partition identified by **0.0.0.1**, the entry in the uniqueness table is **0.0.0.1/account 2286 0**. This account is migrated to the logical partition identified by **0.2.0.1**. When the uniqueness table entries are updated to point to the new location, the entry for this account changes from **0.0.0.1/account 2286 0** to **0.2.0.1/account 2286 0**.

6. It migrates the related transient objects (active-session objects and reservation objects) from the source Oracle IMDB Cache to the destination Oracle IMDB Cache.

About the Transient Objects Migrated by the **pin_amt_tt** Utility

The following tables make up the transient objects migrated by the **pin_amt_tt** utility:

- Active-Session objects:
 - **ACTIVE_SESSION_T**
 - **ACTIVE_SESSION_TELCO_GSM_T**
 - **ACTIVE_SESSION_TELCO_GPRS_T**
 - **ACTIVE_SESSION_TELCO_T**

- ACTIVE_SESSION_RESV_LIST_T
- Reservation objects:
 - RESERVATION_T
 - RESERVATION_BALANCES_T

About Account Synchronization with Pipeline Manager after the Migration

When you use the **pin_amt_tt** utility to migrate accounts, you can enable or disable account synchronization in the instance of the Pipeline Manager associated with the BRM database. You do so by enabling or disabling the **pipeline_sync** configuration parameter in the utility's **pin.conf** file.

If you enabled the **pipeline_sync** configuration parameter, then, after the **pin_amt_tt** utility migrates the accounts to the destination logical partition in the Oracle IMDB Cache grid, it generates an **UpdateLogicalPartition** notification event. A message is added in the appropriate database queue for the Pipeline Manager instance associated with the BRM database.

The **pin_amt_tt** utility does not wait for any acknowledgment from the Pipeline Manager. The pipeline for the BRM database is automatically updated with the database information on the migrated accounts.

About the Message Enqueued for the Pipeline

The message enqueued in the database queue for the pipeline contains information about the new destination to which the accounts were migrated and the access details for the destination Oracle IMDB Cache. When you migrate a large number of accounts, the utility breaks up the account entries into multiple messages.

The fields in the message are:

- PIN_FLD_ACCOUNT_TAG, which is a comma-separated list of the POIDs of the migrated objects.
- PIN_FLD_DEST_DATABASE, which identifies the destination logical partition.

For example, the message enqueued for accounts **352690**, **359678**, and **360254**, which were moved to the logical partition identified as **0.2.0.1**, would be as follows:

```
PIN_FLD_DEST_DATABASE STR [0] "0.2.0.1"
PIN_FLD_ACCOUNT_TAG STR [0] "352690,359678,360254"
```

System and Configuration Requirements

You may have completed some of the following requirements needed to run the **pin_amt_tt** utility, as part of the general installation tasks. Verify the following:

- General system requirements. See ["Installing and Configuring BRM for Account Migration"](#).
- Oracle TimesTen 64-bit client software is installed. See *Oracle TimesTen In-Memory Database Installation Guide* for information.
- Installation required to send BRM events to your pipelines for Account Synchronization is complete. See "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*.

- All other configuration relating to Pipeline Manager that would pertain to account migration and synchronization within an Oracle IMDB Cache. See ["Migrating Accounts with the Pipeline Manager Running"](#).
- The **load_pin_uniqueness** utility has been configured correctly. See ["Configuring the load_pin_uniqueness Utility for Oracle IMDB Cache"](#).

Guidelines for Migrating Accounts with the pin_amt_tt Utility

The following points should be noted when you migrate account data using the **pin_amt_tt** utility:

- The destination Oracle IMDB Cache must be in an active state.
- The following operations may fail during the migration process on accounts selected for migration:
 - AAA opcodes
 - Payment
 - Billing
- If you require account synchronization with Pipeline Manager (that is, **pipeline_sync** is set to 1 in **pin.conf** file), then, to avoid load failures in Rated Event loader:
 1. Stop the Batch Controller before the migration process begins.
 2. Start the Batch Controller after the migration process finishes.
- Customer Center should not be operational during the account migration.
Close the Customer Center client application before you start the migration process and reopen it after you verify that the migration was successfully completed.
- Allow sufficient time for the **pin_amt_tt** utility to wait on accounts that have been selected for migration but are currently locked by another application.

Important: Specify the desired waiting time as the value for **LockWait** in the **sys.odbci.ini** data store configuration file you created in **IMDB_home\info**, where **IMDB_home** is the directory in which you installed Oracle IMDB Cache.

For more information, see ["Installing and Configuring a BRM System with IMDB Cache Manager"](#).

About Account Migration Tasks

Migrating accounts includes the following general procedure. When you move accounts from more than one logical partition to a destination logical partition within the same Oracle IMDB Cache grid, there will be a repetition of the procedure with some changes to suit the specific account migration.

Overview of Migration Procedure

The procedure to migrate a set of selected accounts within the same cache grid consists of the following steps:

1. Verify that your migration follows the recommended guidelines. See ["Guidelines for Migrating Accounts with the pin_amt_tt Utility"](#).

2. Verify that you have configured the **pin.conf** file with the required information. See ["Configuring the pin_amt_tt Utility"](#).
3. Start the account migration application. See ["Running the pin_amt_tt Utility"](#).
4. Monitor the migration. See ["Monitoring Account Migration"](#).
5. Handle any failures that occurred during the migration process. See ["Handling Account Migration Failures"](#).
6. Clean up the destination Oracle IMDB Cache of all old and unnecessary objects. See ["Cleaning Up the Oracle IMDB Cache after Migration"](#).

Configuring the pin_amt_tt Utility

BRM provides a default **pin.conf** configuration file in the *BRM_home/sys/amt_tt* directory. Each time you run the **pin_amt_tt** utility, edit the appropriate **pin.conf** file. See ["Creating Configuration Files for BRM Utilities"](#) for more information.

Note: Before you make any changes to the **pin.conf** file, save a backup copy. When editing this file, follow the instructions in each section.

The **pin.conf** file should contain the required information for the following:

- Access information for the Connection Manager.
- Access information about the Oracle Database where all your data is stored.
- Access information for the data store associated with the logical partition where the selected accounts currently reside.
- Access information for the data store associated with the logical partition to which accounts are to be migrated.
- Selection criteria for the set of accounts you are migrating. See ["Providing the Account Selection Criteria"](#) for more information.
- Whether Account Synchronization is enabled.
- The desired logging level and file information.

[Table 36–1](#) provides information on all the parameters used in the **pin.conf** file.

Providing the Account Selection Criteria

The **pin.conf** file provides a set of entries that can be used to provide selection criteria to be used by the **pin_amt_tt** utility in selecting accounts for migration.

Provide a filter for the account selection by providing values for one, some, or all the criteria listed in the file. The utility migrates only those accounts that fulfill all the criteria you provide in the **pin.conf** file. The **pin_amt_tt** utility requires a value to be set for at least one of the account selection criteria in the configuration file.

By default, all the available account selection criteria are commented out in the **pin.conf** file. Uncomment the entry for each criteria that you require and provide a value for it in the **pin.conf** file.

Configuration File Parameters

[Table 36–1](#) shows the parameters used in defining the **pin.conf** configuration file.

Table 36–1 *pin.conf* Values used by *pin_amt_tt*

Entry	Value	Description
account_status	String	Specifies the status for the accounts that would be fetched for migration. The default is Active . For example: <pre>- pin_amt_tt account_status Active</pre>
bill_day_of_month	Date	Specifies the billing DOM. Accounts with this billing DOM are fetched for migration. The default is 1 . For example: <pre>- pin_amt_tt bill_day_of_month 1</pre>
cm_ptr	String	Specifies the connection information of the CM process.
end_creation_date	Date	Specifies the end date for the range of account creation dates to be used as selection criteria. Enter as MM/DD/YYYY. For example: <pre>- pin_amt_tt end_creation_date 01/31/2011</pre>
logfile	String	Specifies the path to the log file for the sample application.
loglevel	Integer	Specifies the level for the log: <ul style="list-style-type: none"> 0. No logging 1. Log error messages only. This is the default. 2. Log error messages and warnings 0. Log error, warning, and debug messages
login_name	String	Specifies the login name to use to connect to CM process.
login_pw	String	Specifies the password to log in to the Connection Manager (CM) process.
login_type	Integer 0 or 1	Specifies whether the login name and password are required. The value for this entry can be: <ul style="list-style-type: none"> 0. Only a user ID is required 1. Both user name and password are required. This is the default.
max_accounts	Positive integer	Specifies the maximal number of accounts to include in the selection. For example: <pre>- max_accounts 100</pre>
oracle_db_pwd	String	Mandatory parameter. Specifies the user password for logging into an Oracle database.
oracle_db_sid	String	Mandatory parameter. Specifies the Oracle Service ID for logging into an Oracle database.
oracle_db_user	String	Mandatory parameter. Specifies the user name for logging into an Oracle database.

Table 36–1 (Cont.) pin.conf Values used by pin_amt_tt

Entry	Value	Description
pipeline_sync	Integer 0 or 1	Specifies whether synchronization between Pipeline and AMT_TT is needed or not. When set to 1 , the UpdateLogicalPartition notification is created after the migration process is complete. The default is 0 (synchronization is not required). For example: - pin_amt_tt pipeline_sync 0
poid_list	string	A comma-separated list of account POID values. The maximum number of POID entries allowed in this list is 1000. For example: - poid_list 441837,441838
retry_count	Integer	Specifies the number of attempts for the Load , Unload , or UpdateUniqueness attempts. The default is 3 . For example: - pin_amt_tt retry_count 3
source_tt_node_db_id	String	Mandatory parameter. Specifies the database number for a source Oracle IMDB Cache (data store). For example: - pin_amt_tt source_tt_node_db_id 0.0.0.1
source_tt_node_dsn	String	Mandatory parameter. Specifies the data store name for a source Oracle IMDB Cache (data store). For example: - pin_amt_tt source_tt_node_dsn tt_0.0.0.1
source_tt_node_pwd	String	Mandatory parameter. Specifies the user password for logging into a source Oracle IMDB Cache (data store). For example: - pin_amt_tt source_tt_node_pwd pinXX
source_tt_node_user	String	Mandatory parameter. Specifies the user name for logging into a source Oracle IMDB Cache (data store). For example: - pin_amt_tt source_tt_node_user pinXX
start_creation_date	Date	Specifies the start date for the range of account creation dates to be used as selection criteria. Enter as MM/DD/YYYY. For example: - pin_amt_tt start_creation_date 01/01/2011
target_tt_node_db_id	String	Mandatory parameter. Specifies the database number for a destination Oracle IMDB Cache (data store).

Table 36–1 (Cont.) *pin.conf* Values used by *pin_amt_tt*

Entry	Value	Description
target_tt_node_dsn	String	Mandatory parameter. Specifies the data store name for a destination Oracle IMDB Cache (data store).
target_tt_node_pwd	String	Mandatory parameter. Specifies the user password for logging into a destination Oracle IMDB Cache (data store).
target_tt_node_user	String	Mandatory parameter. Specifies the user name for logging into a destination Oracle IMDB Cache (data store).
Userid	String	Specifies the database number and service type for the Portal database.

Running the *pin_amt_tt* Utility

To run the *pin_amt_tt* utility

1. Go to the *BRM_home/sys/amt_tt* directory where the *pin.conf* file is located.
2. Enter the following command.

```
pin_amt_tt
```

The account migration process begins.

Monitoring Account Migration

The *pin_amt_tt* utility makes log entries in its log file. Access the log file for this utility and monitor the account migration to verify that:

- The cache groups were unloaded from the source Oracle IMDB Cache.
- The accounts were migrated to the destination Oracle IMDB Cache.
- The transient objects were migrated to the destination Oracle IMDB Cache (data store). See ["About the Transient Objects Migrated by the *pin_amt_tt* Utility"](#) for information on the tables migrated by this utility.
- The data from the BRM database was loaded into the cache groups located in the destination Oracle IMDB Cache grid node.

Handling Account Migration Failures

The actions BRM takes depend on where the failure occurs in the migration process:

- If there are any problems in migrating the accounts data, the migration changes are rolled back and your system is restored to the state before the migration.

Check the log file and take the necessary actions before you attempt the migration again. The log file is in the directory specified by the **logfile** entry in the *pin.conf* file. See [Table 36–1](#).

- If the migration fails for the active-session objects associated with any of the selected accounts, BRM rolls back the migration of all the active-session objects. It does not try to migrate the reservation objects.

Manually migrate the required active-session and reservation objects for the required accounts. Use the POID information in the log file to identify these objects.

- If the migration succeeds for the active-session objects, but fails for the reservations objects associated with any of the selected accounts, BRM rolls back the migration of all the reservation objects.

Manually migrate the reservation objects for the required accounts. Use the POID information in the log file to identify these objects.

Migrating the Transient Objects Manually

Complete the following steps to migrate the transient objects for the required accounts:

1. Use the POID information in the log file to identify the appropriate transient objects for the accounts.
2. Use SQL statements to migrate those transient objects to the destination Oracle IMDB Cache.

See "[About the Transient Objects Migrated by the `pin_amt_tt` Utility](#)" for the set of transient objects migrated by the `pin_amt_tt` utility.

Cleaning Up the Oracle IMDB Cache after Migration

When you run the `pin_purge` utility on the Oracle IMDB Cache, the cache gets cleared of all unnecessary data objects. However, these objects remain archived in the BRM database.

After the migration, when the `pin_amt_tt` utility loads all the entries for the migrated accounts from the BRM database into the Oracle IMDB Cache, the cache groups for the migrated accounts include the archived data.

To clear up the unnecessary information on the migrated accounts from the Oracle IMDB Cache, use the `pin_purge` utility. See "[pin_purge](#)" for more information.

Modifying Applications to Work with AMM

This chapter provides information on modifying custom client applications and custom reports to work with the Oracle Communications Billing and Revenue Management (BRM) Account Migration Manager (AMM) software. It also provides the list of AMM return codes that are used for automating AMM.

This chapter is for system administrators, database administrators, and programmers. Modifying applications for AMM requires knowledge of the following:

- BRM error handling
- BRM database schema
- BRM opcodes
- BRM Reports

Modifying Custom Client Applications for AMM

Custom client applications that connect to a specific database schema and try to access an object based on a POID may receive a `PIN_ERR_INVALID_OBJ` error if the object was migrated to another database schema. You must modify any custom client applications to handle that error and then perform a global search to find the object's correct location.

To obtain the correct POID of a storable object, modify your application to call the `PCM_OP_GLOBAL_SEARCH` opcode from its exception handling routine.

This example shows a call to the `PCM_OP_GLOBAL_SEARCH` opcode when the `PIN_ERR_INVALID_OBJ` error is returned from the Oracle DM:

```
/* Error? */

if (PIN_ERR_IS_ERR(ebufp)) {
    PIN_ERR_LOG_EBUF(PIN_ERR_LEVEL_ERROR,
        "sample_read_obj_search error", ebufp);
}
/* Call the DM to do a global search.*/

PCM_OP(ctxp, PCM_OP_GLOBAL_SEARCH, 0, flistp, &r_flistp, ebufp);

return;
```

The following opcodes return the `PIN_ERR_INVALID_OBJ` error when a POID specified in an input flist is invalid:

- `PCM_OP_READ_OBJ`

- PCM_OP_READ_FLDS
- PCM_OP_WRITE_FLDS
- PCM_OP_INC_FLDS
- PCM_OP_DELETE_OBJ
- PCM_OP_DELETE_FLDS
- PCM_OP_TRANS_OPEN

Modifying Custom BRM Reports for AMM

After account migration, any custom BRM reports created before Infranet Release 6.2 ServicePak1 might retrieve and process duplicate data from your source and destination database schemas. For example, if an account object is migrated from database schema **0.0.0.1** to database schema **0.0.0.2**, your report might retrieve the account object from both database schemas.

To prevent this, use the Oracle Business Intelligence Publisher to add the following line to the WHERE clause of each custom report's query:

```
TABLE_T.POID_DB > 0
```

where *TABLE_T* satisfies these conditions:

- It is a database table used by the report.
- It is one of the tables moved from the source database schema to the destination database schema when account data is migrated.
- It is associated with every record the report must retrieve.

Note: If a single table does not satisfy the last condition, add the same line for several tables that together satisfy the last condition.

AMM Return Codes and Messages

AMM uses the return codes and messages shown in [Table 37–1](#). To automate account migration, you can modify your external application to check for the following return codes and respond appropriately.

Table 37–1 *AMM Return Codes and Messages*

Return Code Number	Return Code	Return Message
100	CONTROLLER_STARTED_SUCC	controller is started
101	CONTROLLER_STOPPED_SUCC	controller is stopped
102	CONTROLLER_PAUSED_SUCC	paused controller
103	CONTROLLER_CONTINUED_SUCC	continued controller
104	CONTROLLER_UP_SUCC	controller status is up
105	CONTROLLER_DOWN_SUCC	controller status is down
106	SUBMIT_JOB_SUCC	submitted job
107	DELETE_JOB_SUCC	deleted job

Table 37–1 (Cont.) AMM Return Codes and Messages

Return Code Number	Return Code	Return Message
108	PURGE_DATABASE_SUCC	purged database
109	ENABLE_JOB_SUCC	enabled job
110	REPORT_SUCC	generated report
111	ENABLE_BATCH_SUCC	enabled batch
200	CONTROLLER_RUNNING_ERROR	ERROR: controller is already running
201	CONTROLLER_PAUSED_ERROR	ERROR: controller is already paused
202	CONTROLLER_SPEC_ACCESS_ERROR	ERROR: controller specification does not exist
203	CONTROLLER_COMM_ERROR	ERROR: controller cannot be reached
204	SEARCH_SPEC_IO_ERROR	ERROR: account search specification could not be accessed
205	OPERATION_ROLLBACK_ERROR	ERROR: operation rollback
206	SEARCH_SPEC_PARSE_ERROR	ERROR: account search specification cannot be parsed
207	OPERATION_PERM_ERROR	ERROR: operation not permitted for current user OR job_id/batch_id does not exist
208	CONTROLLER_UNKNOWN_HOST_ERROR	ERROR: controller host not found
209	CONTROLLER_PROCESS_ERROR	ERROR: controller process could not be created
210	REPORT_PROCESS_ERROR	ERROR: external process interruption
211	REPORT_SCRIPT_ACCESS_ERROR	ERROR: reporting tool not found or report type does not exist
212	OPT_PARAM_REQ_ERROR	ERROR: one optional parameter is required
213	CONFIG_FILE_ACCESS_ERROR	ERROR: configuration file cannot be accessed
214	INIT_ERROR	ERROR: could not create new object
215	EMPTY_RESULTSET_ERROR	ERROR: account search resulted in 0 accounts, job submission failed
216	CONVERSION_CLASS_LOAD_ERROR	ERROR: dynamic loading of custom Conversion class failed

Modifying the Account Migration Manager

This chapter describes how to create custom search criteria for the Oracle Communications Billing and Revenue Management (BRM) Account Migration Manager (AMM).

This document is for system administrators, database administrators, and programmers. Modifying AMM requires knowledge of the following:

- Java programming language
- PL/SQL
- BRM database schema

Creating Custom Account Search Criteria

AMM enables you to migrate accounts that meet custom criteria. For example, you can create custom criteria for finding and migrating accounts located in a certain state or belonging to a particular service provider.

To create a custom search criteria, perform these tasks:

1. [Creating a Search Template](#)
2. [Adding New Entries to the Account Search Configuration File](#)
3. [Implementing and Compiling the Conversion Interface](#)
4. [Verifying Your Search Criteria](#)

Creating a Search Template

AMM searches for accounts in a database schema by using SQL statements generated from an account search template. Before AMM can generate a SQL statement with new search criteria, you must first create a template for it in the custom account search properties file.

To create a template for your search criteria:

1. Open the custom account search properties file (*BRM_home/apps/amt/com/portal/amt/custom_account_search.properties*) in a text editor.
2. Add SQL fragments for your search criteria by using the following syntax:

```
criteria_name=AND SQL_condition \n
```

Where:

- *criteria_name* is the name of your selection criteria.
- *SQL_condition* is a valid SQL *condition* that searches a BRM table and references one or more search variables, as shown below. Search variables must be surrounded by curly braces "{ }" and match an entry in the **account_search.cfg** file.

```
condition_text '{SearchVariable}'...
```

Important: *SearchVariable* must use a unique name and must not match one of the BRM-defined search variable names. For the list of BRM-defined search variables, see ["Creating the Account Search Configuration File"](#).

For information on the SQL condition, see your Oracle documentation.

3. Save and exit the file.

Sample Search Template

The following sample search template enables AMM to search for accounts located in a particular state. It tells AMM to search the ACCOUNT_NAME_INFO_T table for objects with the **state** field set to a specified value.

```
# select accounts based on state
cust_acct_search_account_state_constraint=\
AND EXISTS \n\
(SELECT an.obj_id0 FROM account_nameinfo_t an \n\
WHERE an.obj_id0 = a.poid_id0 and an.state = '{account_state}') \n
```

Adding New Entries to the Account Search Configuration File

When building a query, AMM replaces the search variables in your account search template with values from the account search configuration file (*BRM_home/apps/amt/account_search.cfg*).

To add an entry for your search variable:

1. Open the *BRM_home/apps/amt/account_search.cfg* file in a text editor.
2. Add your new search entry and comments to the file.

Important: *SearchVariable* must match the search variable name referenced in the **custom_account_search.properties** file.

```
# - You should add comments about the new search entry and
#   valid values.
```

```
SearchVariable=
```

3. Save and exit the file.

Sample Account Search Configuration File

A sample search entry for the **account_state** search criteria:

```
# - Migrates accounts located in a specific state. Valid values
# are California and Oregon.
account_state=
```

Implementing and Compiling the Conversion Interface

Each custom search variable must have a corresponding Java implementation of the **Conversion** interface.

1. Run the appropriate profile script for your shell. This script sets your CLASSPATH and PATH environment variables to the appropriate values. For example, for the c shell:

```
% cd BRM_home/apps/amt
% source profile.csh
```

2. Create a class that implements the **Conversion** interface.
3. Save and compile your *SearchVariable.java* source file in the *BRM_home/apps/amt/com/portal/amt* directory.

```
% cd BRM_home/apps/amt/com/portal/amt
% javac SearchVariable.java
```

This creates a *SearchVariable.class* file in the same directory.

Important: For AMM to successfully build a search with your custom search criteria:

- The class name must match the search variable name used in the **custom_account_search.properties** and **account_search.cfg** files.
 - The class must reside in the *BRM_home/apps/amt/com/portal/amt* directory.
-

Sample Class Implementing Conversion Interface

The following sample class, **account_state.class**, enables users to search for accounts from California or Oregon.

```
package com.portal.amt;
public class account_state implements Conversion {
    public String convert(String stateName) throws ConversionException {
        String stateCode = null;
        if(stateName.equals("California")) {
            stateCode = "CA";
        } else if(stateName.equals("Oregon")) {
            stateCode = "OR";
        } else {
            throw new
                ConversionException("Error: account_state " + stateName + " unknown.");
        }
        return(stateCode);
    }
}
```

Verifying Your Search Criteria

Before migrating accounts with the new search criteria, verify its accuracy by:

1. [Verifying That the Search Criteria Creates Valid SQL Statements](#)
2. [Verifying That the Search Criteria Finds Correct Accounts](#)

Verifying That the Search Criteria Creates Valid SQL Statements

Use the "[pin_amt_test](#)" utility to verify that your custom search template generates a valid SQL statement.

1. Open your account search configuration file (*BRM_home/apps/amt/account_search.cfg*) in a text editor.
2. Enter values for the source and destination database schemas, the batch size, and your custom search criteria.

For example, you might enter the following to test the **account_state** criteria:

```
src_database=0.0.0.1
dest_database=0.0.0.2
start_creation_date=
end_creation_date=
product_name=
account_status=
bill_day_of_month=
max_accounts=
batch_size=50
poid_list=
account_state=California
```

3. Save and exit the file.
4. Use the **pin_amt_test** utility to generate a SQL statement with your new search criteria.

```
% pin_amt_test -c AccountSearchFile
```

If successful, the utility displays the resulting SQL statement. For example, the sample **account_state** criteria generates the following:

```
Compile: account_search.cfg
-----
account search SELECT statement:
-- acct_search_select: default
SELECT
DISTINCT a.poid_id0
FROM account_t a
WHERE
...
```

If the compilation failed, the utility returns the file name and line number where the error occurred. For example, the utility returns the following when users enter an invalid state for the sample **account_state** criteria:

```
compile: account_search.cfg
-----
account_search.cfg:32: mapping of account_state field value Florida failed
```

5. Verify that the resulting SQL statement is correct.

Verifying That the Search Criteria Finds Correct Accounts

Use the **pin_amt_test** utility to verify that your search criteria works properly. This utility only displays results on the screen and does not migrate your objects.

To verify your search query:

1. Create a database schema with a precisely defined set of account data. The database schema should contain a small number of accounts.

2. Create a list of account POIDs that meet the custom criteria you are testing. For example, write down the POIDs of all accounts created in California.
3. Open your account search configuration file (*BRM_home/apps/amt/account_search.cfg*) in a text editor.
4. Enter values for the source and destination database schemas, the batch size, and your custom search criteria.

For example, you might enter the following to test the **account_state** criteria:

```
src_database=0.0.0.1
dest_database=0.0.0.2
start_creation_date=
end_creation_date=
product_name=
account_status=
bill_day_of_month=
max_accounts=
batch_size=50
poid_list=
account_state=California
```

5. Save and exit the file.
6. Use the **pin_amt_test** utility to execute your search query against the source database schema:

```
% pin_amt_test -e AccountSearchFile
```

7. The utility prints to the screen a list of account POIDs that meet your search criteria. Compare this list of POIDs with the list you created in step 2.

If the lists match, your new search criteria works properly and you can start using it to migrate accounts.

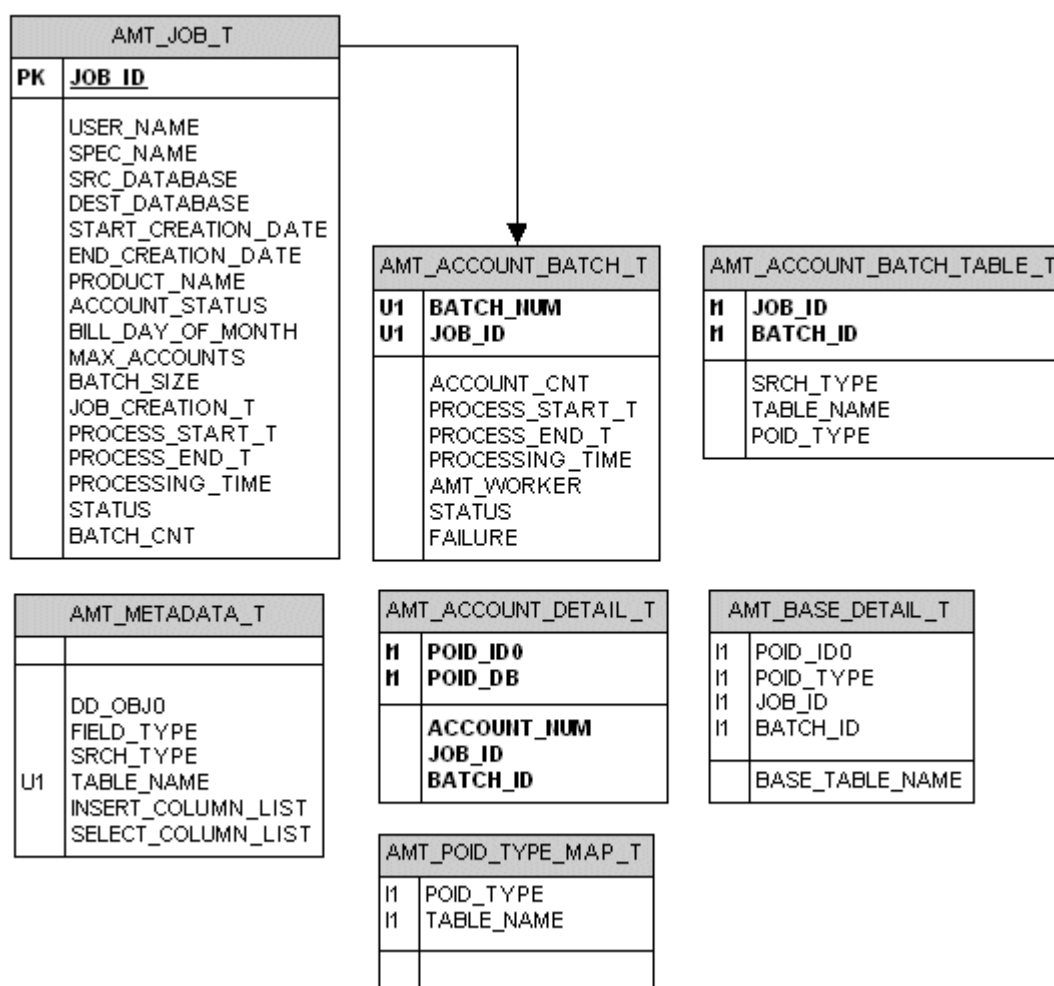
If the lists do not match, make sure:

- Your search template generates a valid SQL statement.
- Your search template, search configuration file, and class all refer to the same variable name.

AMM Entity Relationship Diagram

Figure 39–1 shows the Account Migration Manager Entity Relationship (ER) Diagram.

Figure 39–1 Account Migration Manager ER Diagram



Account Migration Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Account Migration utilities.

pin_amt

Use this utility to migrate accounts from a source database schema to a destination database schema in the same BRM database.

When you use Oracle IMDB Cache in your environment, this utility also migrates account objects from the source Oracle IMDB Cache (associated with the source BRM database schema) to the destination Oracle IMDB Cache (associated with the destination BRM database schema).

You define which accounts to migrate in the account search configuration file in the *BRM_home/apps/amt* directory. See ["Creating the Account Search Configuration File"](#).

Note: To connect to the BRM database, the **pin_amt** utility needs a configuration file in the *BRM_home/sys/amt* directory. For information on how to create this file, see ["Connecting AMM to Your Database Schemas"](#).

Location

BRM_home/bin

Syntax

```
pin_amt  [-f ConfigFileName] [-a ControllerID]
          [-c start | stop | pause | continue | status | log]
          [-s AccountSearchFile] [-d JobID]
          [-r list_jobs | job_details | group_details]
          [-p SourceDatabaseSchema] [-e JobID] [-b JobID:BatchNumber] [-h]
```

Parameters

-f ConfigFileName

Specifies the name of the configuration file that defines how to connect to each database schema in your system. By default, the **pin_amt** utility looks in the *BRM_home/sys/amt* directory. If your configuration file is located in a different directory, you must specify the entire path for the file.

If you use the *BRM_home/sys/amt/Infranet.properties* file, you can ignore this parameter.

-a ControllerID

Specifies the AMM Controller to use.

Use this option only with the **-c** option and only when your system contains multiple AMM Controllers. If your system contains only one AMM Controller, ignore this option.

-c start | stop | pause | continue | status | log

Sets the AMM Controller. When your system contains multiple AMM Controllers, you must also use the **-a** option.

Use one of these options with the parameter:

-c start starts the AMM Controller.

-c stop stops the AMM Controller. If you stop the AMM Controller while it is processing a batch, it finishes processing the batch before stopping.

-c pause pauses the processing of a job in the queue. If you pause the Controller while it is processing a batch, it finishes processing the batch before pausing.

-c continue restarts processing a job that was paused.

-c status displays the current status of the AMM Controller.

-c log displays all AMM Controller transactions in real time through an Xterm window. To use this option, you must set the **DISPLAY** environment variable correctly.

-s AccountSearchFile

Specifies the name of the configuration file that defines which accounts to migrate. For information on how to create the file, look at the sample account search configuration file (*BRM_home/apps/amt/account_search.cfg*).

By default, the **pin_amt** utility looks in the current working directory. If your configuration file is located in a different directory, you must specify the entire path for the file.

-d JobID

Deletes the specified job from the job management tables. When deleting a job that migrated successfully, this option also purges all migrated accounts from the source database schema.

-e JobID

Enables the specified job in the queue.

-r list_jobs | job_details | group_details

Runs the preconfigured report. Use one of these options with the parameter:

-r list_jobs displays the status of all jobs currently in the queue.

-r job_details displays the details of the specified job.

-r group_details displays the details of the specified account group.

-p SourceDatabaseSchema

Purges all accounts that were successfully migrated from the source database schema. For example, to purge invalid objects from your primary schema, enter the following:

```
pin_amt -p 0.0.0.1
```

-h

Displays the syntax and parameters for this utility.

-b JobID:BatchNumber

Changes the status of the batch from FAILED to NOT PROCESSED, and the job from FINISHED to DISABLED. For information, see ["About Batch Status Flags"](#) and ["About Job Status Flags"](#).

Results

The **pin_amt** utility notifies you when it successfully completes a command.

For error information about each job, run a report or look in the AMM Mover log file. The log file is in the directory specified by the **0.0.0.x_mover_log_file_dir** entry in the **Infranet.properties** file.

For error information about the AMM Controller, look in the AMM Controller log file. The log file is in the directory specified by the **controller_N_log_directory** entry in the **Infranet.properties** file.

The history of all **pin_amt** commands is located in the **pin_amt** log file.

pin_amt_test

Use this utility to test your custom BRM account search criteria. This utility safely executes your search criteria against a source database schema and displays either a SQL SELECT statement or a list of account POIDs meeting your search criteria.

You define which custom search criteria to test in the account search configuration file (*BRM_home/apps/amt/account_search.cfg*). See ["Creating the Account Search Configuration File"](#).

Note: To connect to the BRM database, the **pin_amt_test** utility needs a configuration file in the *BRM_home/sys/amt* directory. For information on how to create this file, see ["Connecting AMM to Your Database Schemas"](#).

Location

BRM_home/apps/amt

Syntax

```
pin_amt_test [-f ConfigFileName ]
              -c AccountSearchFile | -e AccountSearchFile | -h
```

Parameters

-f ConfigFileName

Specifies the name of the configuration file that defines how to connect to each database schema in your system. By default, the **pin_amt_test** utility looks in the *BRM_home/sys/amt* directory. If your configuration file is located in a different directory, you must specify the entire path for the file.

If you use the *BRM_home/sys/amt/Infranet.properties* file, you can ignore this parameter.

-c AccountSearchFile

Displays the SQL SELECT statement generated with the account search criteria specified in *AccountSearchFile*.

By default, the **pin_amt_test** utility looks in the current working directory. If your account search file is located in a different directory, you must specify the entire path for the file.

-e AccountSearchFile

Executes the SQL SELECT statement for the specified search criteria against the source database schema and displays the list of account POIDs meeting the search criteria.

By default, the **pin_amt_test** utility looks in the current working directory. If your account search file is located in a different directory, you must specify the entire path for the file.

-h

Displays the syntax and parameters for this utility.

Results

The **pin_amt_test** utility prints to the screen the SQL SELECT statement, a list of accounts meeting your search criteria, or an Oracle error message. For information about Oracle error messages, see your Oracle documentation.

pin_amt_tt

Use this utility to migrate accounts from one logical partition to another logical partition in the same Oracle IMDB Cache grid.

You define which accounts to migrate in the account search configuration file in the *BRM_home/apps/amt_tt* directory. See "[Configuring the pin_amt_tt Utility](#)".

Location

BRM_home/bin

Syntax

pin_amt_tt]

Parameters

None

Results

For information about the migration, check the **pin_amt_tt** log file. The log file is in the directory specified in the **pin.conf** file.

Part IX

Running Business Operations

Part IX describes how to configure Oracle Communications Billing and Revenue Management Business Operations Center. It contains the following chapters:

- [Using Business Operations Center](#)
- [Business Operations Center Utilities](#)

Using Business Operations Center

This chapter provides guidelines to help you manage Oracle Communications Billing and Revenue Management Business Operations Center.

Before you read this chapter, 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 must 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.
- **Collecting payments:** Collects the balance due for accounts that use 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.
- **Synchronizing product catalogs:** Synchronizes the catalogs stored in the BRM server with updated values from customers.
- **Refunding payments:** Finds accounts that have refund items, and makes online refund transactions.
- **Tracking business trends:** Shows business trends based on data generated by the preceding operations.

Business Operations Center calls the internal `pin_job_executor` utility to run the BRM utilities for these operations.

For more information about Business Operations Center, see the Business Operations Center Help.

About Performing Operations in Business Operations Center

You can configure and run only the following operations in Business Operations Center: billing, payment collections, invoicing, general ledger reports, product catalog synchronization, and refunds. Oracle recommends the following:

- Because the operations that BOC can run often depend on other processes that BOC cannot run, determine whether you must supplement some of the operations run in BOC by running additional processes elsewhere in BRM.

For example, to process off-cycle (daily, weekly, quarterly, and so on) charges, you must run the **pin_cycle_fees** utility, which is not supported by Business Operations Center, from the BRM command line or 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 use something other than Business Operations Center to 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.
- Do not use best pricing. It is incompatible with Business Operations Center.

For more information about performing operations in Business Operations Center, see the Business Operations Center Help.

Enabling Secure Communication for 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.

Important: 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=wallet_location/wallet`

where *wallet_location* is the full path to the directory in which your Oracle wallet resides.
 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.

Rendering Invoices in a Third-Party Invoice Application

By default, when Business Operations Center runs the internal `pin_job_executor` utility to generate invoices, the invoices are generated and rendered using Oracle Business Intelligence (BI) Publisher. If you use a third-party invoice application, you must configure the `pin_job_executor` utility to ensure that the generated invoices are not rendered in BI Publisher.

To render invoices in a third-party application:

1. Open the `BRM_home/apps/pin_job_executor/Infranet.properties` file in a text editor.

Important: 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`

The `pin_job_executor` utility will not render the invoices in BI Publisher.

4. 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

To generate business metrics data:

1. Go to the `BRM_home/apps/pin_generate_analytics` directory.
2. Run the following command:

```
pin_generate_analytics[-parameter]
```

where *parameter* is one of the following:

- **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.
- **billing** generates business metrics data about the amount billed and the number of bills generated.
- **acc_recv** generates business metrics data about the A/R for the last one year with respect to month and the year.

Note: If you run this utility without parameters, data will be generated for all operations: subscribers, payment, billed revenue, and A/R.

For more information, see "[pin_generate_analytics](#)".

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/password@database_alias
```

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
```

Business Operations Center Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) utilities for Oracle Communications Billing and Revenue Management Business Operations Center.

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.

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, 2016, A/R business metrics data are generated for a period starting from March 17, 2015 until March 17, 2016.

Results

The progress of the program is displayed on the screen.

pin_job_executor

Business Operations Center uses this utility to run BRM applications such as **pin_bill_accts**, **pin_collect**, and **pin_inv_accts**.

Important: The **pin_job_executor** utility is run internally by Oracle Communications Billing and Revenue Management Business Operations Center. Do not run this utility from the command line.

Part X

Configuration File Reference

Part IX 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](#)

Business Logic pin.conf Reference

This chapter lists the business logic **pin.conf** settings used for configuring Oracle Communications Billing and Revenue Management (BRM).

See also ["business_params Reference"](#) and ["System Administration pin.conf Reference"](#).

Accounts Receivable pin.conf Entries

[Table 43–1](#) lists the Accounts Receivable **pin.conf** entries.

Table 43–1 Accounts Receivable pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
balance_coordinator	fm_bal	Specifies how BRM tracks the total resources reserved by a balance group. See "Configuring How BRM Calculates Reservation Balances" in <i>BRM Telco Integration</i> .	CM	This value is read by the utility when it runs. You do not need to restart the CM.
calc_cycle_from_cycle_start_t	fm_bill	Specifies whether to calculate product fees based on the product's purchase date (PIN_FLD_CYCLE_START_T). See "Calculating Product Cycle Fees for Backdating" in <i>BRM Configuring and Running Billing</i> .	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 parent and its nonpaying child accounts or are consolidated into a single tax item for both the parent and child accounts. See "Specifying How to Calculate Taxes" in <i>BRM Calculating Taxes</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–1 (Cont.) Accounts Receivable pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
item_search_batch	fm_bill	Specifies the number of items returned by a step search. See "Improving Item Search Performance" in <i>BRM Managing Accounts Receivable</i> .	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 because of euro and Economic and Monetary Union (EMU) conversions. See "Rounding Errors for Overpayments and Underpayments" in <i>BRM Managing Customers</i> .	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 because of euro and EMU conversions. See "Rounding Errors for Overpayments and Underpayments" in <i>BRM Managing Customers</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Billing pin.conf Entries

Table 43–2 lists the Billing pin.conf entries.

Table 43–2 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 <i>BRM Configuring and Running Billing</i> .	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 billing type. See "Setting the Default Accounting Type" in <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–2 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Running Billing</i> .	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. See "Customizing How to Bill Events That Occur between Billing Cycles" in <i>BRM Configuring and Running Billing</i> .	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Running Billing</i> .	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
cycle_delay_align	fm_bill	Specifies whether to align the product purchase, cycle, and usage start and end times to the accounting cycle. See "Aligning Account and Cycle Start and End Times" in <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–2 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
cycle_delay_use_special_days	fm_bill	Sets the delayed cycle start date to the 1st of the following month for all deals purchased on the 29th, 30th, or 31st. See "Setting Delayed Cycle Start Dates to the 29th, 30th, or 31st" in <i>BRM Configuring and Running Billing</i> .	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 a Deadlock".	pin_bill	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 <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
delta_step	pin_bill	Reduces contention at the database level during billing. See "Rearranging Accounts to Improve Billing Performance".	pin_bill	This entry is read by the utility when it runs. You do not need to restart the CM.
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 <i>BRM Configuring and Running Billing</i> .	pin_bill	This entry is read by the utility when it runs. You do not need to restart the CM.
enforce_scoping	cm	Turns off branding conditions during billing. See "Activating Branded Service Management" in <i>BRM Managing Customers</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
keep_cancelled_products_or_discounts	fm_subscription_pol	Specifies whether to keep canceled products and discounts. See "Providing Discounts to Closed Accounts" in <i>BRM Configuring and Running Billing</i> .	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. See "Setting the Thresholds That Trigger Automatic Rerating" in <i>BRM Setting Up Pricing and Rating</i> .	CM	The new value becomes effective immediately. You do not need to restart the CM.

Table 43–2 (Cont.) Billing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry
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 <i>BRM Configuring and Running Billing</i> .	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. See "Rounding Up Long Billing Cycles" in <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
stop_bill_closed_accounts	fm_bill	Stops billing of closed /billinfo objects when all items have zero due. See "Suspending Billing of Closed Accounts" in <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
unset_error_status	pin_bill_accts	Sets billing status in the /billinfo object when billing errors occur. See "Setting the Bill Unit Status When Billing Errors Occur" in <i>BRM Configuring and Running Billing</i> .	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 product is purchased or canceled. See "Calculating Prorated Cycle Fees" in <i>BRM Configuring and Running Billing</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Collections pin.conf Entries

Table 43–3 lists the Collections **pin.conf** entries.

Table 43–3 Collections pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
collections_action_dependency	fm_collect	Creates dependencies between collections actions in a collections scenario. See "Creating Dependencies between Collections Actions in a Scenario" in <i>BRM Collections Manager</i> .	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 <i>BRM Collections Manager</i> .	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 <i>BRM Collections Manager</i> .	pin_collections pin_inv	The new value is effective immediately. You do not need to restart the CM.
email_option	pin_collections_send_dunning pin_inv_send	Specifies whether to attach the invoice or include it in the email body. See "Setting the Email Delivery Preference" in <i>BRM Collections Manager</i> .	pin_inv pin_collections	The new value is effective immediately. You do not need to restart the CM.
execute_all_actions	fm_collections	Specifies whether to execute all actions, including actions due before the date on which pin_collections_process is run for the first time, or only to execute actions due after the date on which pin_collections_process is run for the first time.	pin_collections	The new value is effective immediately. You do not need to restart the CM.

Customer Management pin.conf Entries

Table 43–4 lists the Customer Management **pin.conf** entries.

Table 43–4 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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Managing Customers</i> .	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 enables a customer to confirm account creation. See "Changing the Introductory Message Location" in <i>BRM Managing Customers</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

Discounting pin.conf Entries

Table 43–5 lists the Discounting **pin.conf** entries.

Table 43–5 Discounting pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
cdc_line_cancel_day_include	fm_subscription	Specifies whether the day that the subscription was canceled is deducted from contract days. See "Specifying Whether to Count the Days on Which Subscription Status Changes" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

Table 43–5 (Cont.) Discounting pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
cdc_line_create_day_include	fm_subscription	Specifies whether to include or exclude the days on which the subscription service status changes. See "Specifying Whether to Count the Days on Which Subscription Status Changes" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
non_currency_cdc	fm_subscription	Counts aggregated contract days for billing time discount. See "Configuring BRM to Track the Number of Contract Days" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
non_currency_mfuc	fm_subscription	Counts aggregated monthly fee and usage. See "Configuring BRM to Track Monthly Fees and Usage" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
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 <i>BRM Managing Accounts Receivable</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
rollover_zeroout_discounts	fm_rate	Zeroes-out the positive bucket. See "Configuring the Start and End Times for Discount Sharing" in <i>BRM Managing Accounts Receivable</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
time_stamp_cdc	fm_subscription	Specifies whether the day of service reactivation is included in contract days. See "Specifying Whether to Count the Days on Which Subscription Status Changes" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

General Ledger pin.conf Entries

Table 43–6 lists the General Ledger **pin.conf** entries.

Table 43–6 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 <i>BRM Collecting General Ledger Data</i> .	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 <i>BRM Collecting General Ledger Data</i> .	pin_billd	The new value is effective immediately. You do not need to restart the CM.

Invoicing pin.conf Entries

Table 43–7 lists the Invoicing **pin.conf** entries.

Table 43–7 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: <ul style="list-style-type: none"> ▪ pin_inv_accts ▪ pin_inv_export ▪ pin_inv_send ▪ pin_inv_upgrade See "Configuring the Invoice pin.conf for Multiple Database Schemas" in <i>BRM Designing and Generating Invoices</i> .	pin_inv	The new value is effective immediately. You do not need to restart the CM.
event_cache	fm_inv	Enables the event cache for the PIN_FLD_BAL_IMPACTS array. See "Specifying Event Fields to Cache for Invoicing" in <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	pin_inv	The new value is effective immediately. You do not need to restart the CM.

Table 43–7 (Cont.) Invoicing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
from	pin_inv_send	Specifies the e-mail address of the sender. See "About Invoices" and "pin_inv_send" in <i>BRM Designing and Generating Invoices</i> .	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. See "Using HTML Invoice Templates" in <i>BRM Designing and Generating Invoices</i> .	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. see "improving performance when generating invoices" in <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	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. See "Improving Performance by Removing Invoice Details You Do Not Need" in <i>BRM Designing and Generating Invoices</i> .	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. See "Changing the Welcome Message Sender Address" in <i>BRM Managing Customers</i> .	CM pin_inv	The new value is effective immediately. You do not need to restart the CM.

Table 43–7 (Cont.) Invoicing pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
service_centric_invoice	fm_inv_pol	Enables service-centric invoicing. See "Creating Service-Centric Invoices" in <i>BRM Designing and Generating Invoices</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
show_rerate_details	fm_inv_pol	Specifies whether to display shadow adjustment details on invoices. See "Including Shadow Event Adjustment Details in Invoices" in <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	pin_inv	The new value is effective immediately. You do not need to restart the CM.

Payments pin.conf Entries

Table 43–8 lists the Payments **pin.conf** entries.

Table 43–8 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 registration. See "Charging Customers at Registration" in <i>BRM Managing Customers</i> .	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. See "Revalidating Credit Cards" in <i>BRM Managing Customers</i> .	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 registration. See "Validating Credit Cards at Registration" in <i>BRM Managing Customers</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

Table 43–8 (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. See "Creating a New /config/payment Object" in <i>BRM Managing Customers</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	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 registration. See "Enabling Direct Debit Processing" in <i>BRM Configuring and Collecting Payments</i> .	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. See "Enabling Direct Debit Processing" in <i>BRM Configuring and Collecting Payments</i> .	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 registration. See "Enabling Direct Debit Processing" in <i>BRM Configuring and Collecting Payments</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	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 <i>BRM Configuring and Running Billing</i> .	pin_billd	The new value becomes effective immediately. You do not need to restart the CM.

Table 43–8 (Cont.) Payments pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Configuring and Running Billing</i> .	CM	The new value becomes effective immediately. You do not need to restart the CM.
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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
validate_acct	fm_pymt_pol	Enables 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 <i>BRM Managing Customers</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

Pricing and Rating pin.conf Entries

Table 43–9 lists the Pricing and Rating **pin.conf** entries.

Table 43–9 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 . See "Configuring Automatic Rerating of Backdated Events" in <i>BRM Setting Up Pricing and Rating</i> .	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. See "Configuring Automatic Rerating of Backdated Events" in <i>BRM Setting Up Pricing and Rating</i> .	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 Price Lists ".	CM	Cached by the CM. Restart the CM after changing this entry.
cancel_tolerance	fm_bill	Specifies the cancellation tolerance of account products, in minutes. See "Canceling Products without Charging a Cancel Fee" in <i>BRM Managing Customers</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
delay	pin_rerate	Specifies the delay for rerating jobs. See "Specifying the Event Sequence for Rerating" in <i>BRM Setting Up Pricing and Rating</i> .	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. See "Setting Optional Rating Flags" in <i>BRM Setting Up Pricing and Rating</i> .	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 "Updating the pin.conf Configuration File for Offer Profiles" and "Policy-Driven Charging" in <i>BRM Setting Up Pricing and Rating</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–9 (Cont.) Pricing and Rating pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 Price Lists" .	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 product provisioning tags. See "Improving Performance for Loading Large Price Lists" .	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 Price Lists" .	CM	The new value is effective immediately. You do not need to restart the CM.
log_refresh_product	fm_rate	Specifies whether to log price list changes. You can use the log entries to generate notifications about such changes. See "Logging Changes to Price Lists" in <i>BRM Setting Up Pricing and Rating</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
non_currency_linecount	fm_subscription	Counts aggregated number of lines. See "Configuring BRM to Track the Number of Subscriptions" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
rate_change	fm_subscription	Enables the enhanced rate change management feature. See "Configuring BRM to Apply Multiple Rates in a Cycle" in <i>BRM Configuring and Running Billing</i> .	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. See "Specifying a Time Zone" in <i>BRM Setting Up Pricing and Rating</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–9 (Cont.) Pricing and Rating pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
refresh_product_interval	fm_rate	Specifies the interval in which price lists are refreshed in cache memory. See "Setting the Interval for Checking for Price List 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. See "Resetting the Server Time Zone" in <i>BRM Setting Up Pricing and Rating</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
update_interval	fm_zonemap_pol	Specifies an interval for revising the zone map. See "Setting the Interval for Updating Zone Maps" .	CM	Cached by the CM. Restart the CM after changing this entry.
validate_deal_dependencies	fm_utils	Specifies whether to validate the deal prerequisites and mutually exclusive relations. See "Enabling Deal Dependencies Validation" in <i>BRM Managing Customers</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

Publication pin.conf Entries

[Table 43–10](#) lists the publication **pin.conf** entries.

Table 43–10 Publication pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
ece_real_time_sync_db_no	fm_publish	Enables changes to customer data in the BRM database to be published to the ECE cache in real time through External Manager (EM) Gateway. See the discussion about synchronizing BRM and ECE customer data in the ECE documentation.	CM	Cached by the CM. Restart the CM after changing this entry.

Registration pin.conf Entries

[Table 43–11](#) lists the Registration **pin.conf** entries.

Table 43–11 Registration 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 /billinfo is inactive. See "Allowing Active Services with Inactive Accounts" in <i>BRM Managing Customers</i> .	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. See "Disabling the Credit Card Checksum" in <i>BRM Managing Customers</i> .	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.
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. See "Sending Account Information to Your Application When an Account Is Created" in <i>BRM Managing Customers</i> .	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). See "Specifying the Default Country" in <i>BRM Managing Customers</i> .	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. See "Setting the Default Account Currency" in <i>BRM Managing Customers</i> .	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. See "Defining Customer Email Domain Names" in <i>BRM Managing Customers</i> .	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. See "Enabling the Welcome Message" in <i>BRM Managing Customers</i> .	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

Table 43–11 (Cont.) Registration pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
welcome_dir	fm_cust_pol	Specifies the location of the welcome message sent to customers after account creation. See "Specifying the Welcome Message Location" in <i>BRM Managing Customers</i> .	CM	The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

Revenue Assurance pin.conf Entries

[Table 43–12](#) lists the Revenue Assurance **pin.conf** entries.

Table 43–12 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 "Changing the Control Point for Data on Written-Off EDRs" in <i>BRM Collecting Revenue Assurance Data</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Service Lifecycle Management pin.conf Entries

[Table 43–13](#) lists the Service Lifecycle Management **pin.conf** entries.

Table 43–13 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 <i>BRM Managing Customers</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–13 (Cont.) Service Lifecycle Management pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Managing Customers</i> .	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 <i>BRM Managing Customers</i> .	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 <i>BRM Managing Customers</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Services Framework pin.conf Entries

Table 43–14 lists the Services Framework **pin.conf** entries.

Table 43–14 Service Framework pin.conf Entries.

Name	Program	Description	pin.conf File	Implementation
agent_return	fm_tcf	Specifies the provisioning status when provisioning is simulated. See "Using the Network Simulator" in <i>BRM Telco Integration</i> .	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 <i>BRM Telco Integration</i> .	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 <i>BRM Telco Integration</i> .	dm_prov_telco	This entry is read by the utility when it runs. You do not need to restart the CM.

Table 43–14 (Cont.) Service Framework pin.conf Entries.

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Telco Integration</i> .	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 <i>BRM Telco Integration</i> .	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 <i>BRM Telco Integration</i> .	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 product-level provisioning. See "Enabling Wireless Provisioning" in <i>BRM Telco Integration</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
simulate_agent	fm_tcf	Creates a response and updates the service order. See "Using the Network Simulator" in <i>BRM Telco Integration</i> .	CM	The new value is effective immediately. You do not need to restart the CM.
wait_for_all_interim_stop_request	fm_tcf_aaa	Checks for all interim stop accounting requests before processing the final stop accounting request. See "Ensuring That All Subsessions Have Stopped before Closing the Master Session" in <i>BRM Telco Integration</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Tax Calculation pin.conf Entries

Table 43–15 lists the Tax Calculation **pin.conf** entries.

Table 43–15 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 <i>BRM Calculating Taxes</i> .	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 <i>BRM Calculating Taxes</i> .	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 <i>BRM Calculating Taxes</i> .	dm_vertex	Valid only when commtax_series is set to 6 . Restart the DM after changing this entry.
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 <i>BRM Calculating Taxes</i> .	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. See "Defining a Default Ship-From Locale" in <i>BRM Calculating Taxes</i> .	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 <i>BRM Calculating Taxes</i> .	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 "Specifying the Database Server Name, User ID, and Password for Sales Tax Q Series" in <i>BRM Calculating Taxes</i> .	dm_vertex	Restart the DM after changing this entry.

Table 43–15 (Cont.) Tax Calculation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
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 <i>BRM Calculating Taxes</i> .	dm_vertex	Restart the DM after changing this entry.
quantumdb_server	dm_vertex	Specifies the name of the database server that contains the Quantum tables. See "Specifying the Database Server Name, User ID, and Password for Sales Tax Q Series" in <i>BRM Calculating Taxes</i> .	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 "Specifying the Database Server Name, User ID, and Password for Sales Tax Q Series" in <i>BRM Calculating Taxes</i> .	dm_vertex	Restart the DM after changing this entry.
quantumdb_user	dm_vertex	Specifies the name of the Oracle user. See "Specifying the Database Server Name, User ID, and Password for Sales Tax Q Series" in <i>BRM Calculating Taxes</i> .	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). See "Configuring the Default Tax Method for Account-Level Adjustments" in <i>BRM Managing Accounts Receivable</i> .	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. See "Itemizing Taxes by Jurisdiction" in <i>BRM Calculating Taxes</i> .	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. See "Modifying Tax Data after Calculating Taxes" in <i>BRM Calculating Taxes</i> .	CM	Cached by the CM. Restart the CM after changing this entry.

Table 43–15 (Cont.) Tax Calculation pin.conf Entries

Name	Program	Description	pin.conf File	Implementation
tax_reversal_with_tax	fm_ar	Specifies whether to apply a tax reversal for an adjustment, dispute, or settlement See "Configuring Taxes for Adjustments, Disputes, and Settlements" in <i>BRM Calculating Taxes</i> .	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. See "Specifying the Location of the tax_supplier_map File" in <i>BRM Calculating Taxes</i>	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 registration. See "Specifying Whether to Validate ZIP Codes" in <i>BRM Calculating Taxes</i>	CM	The new value is effective immediately. You do not need to restart the CM.
taxation_switch	fm_bill	Enables taxation. See "Enabling Taxation During Real-Time Rating or Billing" in <i>BRM Calculating Taxes</i> .	CM	Cached by the CM. Restart the CM after changing this entry.
taxcodes_map	fm_rate	Specifies the location of the taxcodes_map file. See the following in <i>BRM Calculating Taxes</i> : <ul style="list-style-type: none">■ "Calculating Flat Taxes by Using the taxcodes_map File"■ Vertex: "Specifying the Location of the taxcodes_map File"	CM Billing utilities	Cached by the CM. Restart the CM after changing this entry.
use_charge_to_category_codes	dm_vertex	Specifies the category codes that use ChargeTo as the primary place of use (PPU) location. The default is 10 (wireless). See "Configuring the Vertex DM for Communications Tax Q Series" in <i>BRM Calculating Taxes</i> .	dm_vertex	Restart the DM after changing this entry.
vertex_db	fm_rate	Specifies the database number of the Vertex database. See "Specifying Connection Entries" in <i>BRM Calculating Taxes</i> .	CM	The new value is effective immediately. You do not need to restart the CM.

System Administration pin.conf Reference

This chapter lists the **pin.conf** settings used by system administrators in Oracle Communications Billing and Revenue Management (BRM).

See also ["business_params Reference"](#) and ["Business Logic pin.conf Reference"](#).

Connection Manager (CM) pin.conf Entries

[Table 44–1](#) lists the CM **pin.conf** entries.

Table 44–1 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 <i>BRM Developer's Guide</i> .
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.
cm_logfile	cm	Specifies the full path to the log file used by the CM. See "Changing the Name or Location of a Log File" .
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 from the CM" .
cm_ports	cm	Specifies the port number of the computer where the CM runs. See "About Connecting BRM Components" .

Table 44–1 (Cont.) CM pin.conf Entries

Name	Program	Description
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. See "Activating Branded Service Management" in <i>BRM Managing Customers</i> and "Setting Data Manager Attributes" .
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" .
em_group	cm	Configures the CM to send requests to the NET_EM module. See "Configuring the CM to Send Real-Time Requests to the NET_EM Module" .
em_pointer	cm	Specifies where to find EMs that provide other opcodes to the CM. See "Configuring the CM to Send Real-Time Requests to the NET_EM Module" .
enable_pcm_op_call_stack	dm	Specifies whether PCM_OP_CALL_STACK is printed in the cm.pinlog file after each opcode from a nab (Network Application Program) is completed. See "Recording Opcode Calls in the CM Log File" and "Getting Opcode Statistics from IMDB Cache DM" .
fetch_size	pin_billd pin_mta	Specifies the number of objects received from the database in a block and cached in system memory for processing. See "Tuning the Account Cache Size for Billing Utilities (fetch_size)" and "Tuning the Account Cache Size for Invoice Utilities (fetch_size)" .
fm_offer_profile_cache	cm_cache	Loads all offer_profile objects at initialization of fm_offer_profile library.
fm_utils_content_srvc_profile_cache	cm_cache	Specifies attributes of the content service profile cache. See "Setting Up Content Manager" in <i>BRM Content Manager</i> .
group_children_fetch	fm_bill	Performs a step search for child accounts in groups when the parent group contains many members. See "Filtering Search Results" .
group_members_fetch	fm_bill	Performs a step search for members of the group sharing object when the parent group contains many members. See "Filtering Search Results" .
ip		Specifies the IP address of the devices managed by IP Address Manager. See "Using the IP Address Manager APIs" in <i>BRM Telco Integration</i> .

Table 44–1 (Cont.) CM pin.conf Entries

Name	Program	Description
item_fetch_size	fm_bill	Performs a step search for items. See "Filtering Search Results" .
login_audit	cm	Creates the session event's storable object for each client application. See "Turning Off Session-Event Logging" .
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" and "Getting Opcode Statistics from IMDB Cache DM" .
monitor	pin_mta	Specifies the path and name of a shared memory map file used by the pin_mta_monitor utility. See "Creating Configuration Files for BRM Utilities" .
passwd_age	cm	Specifies the number of days after which the password expires. See "Setting the Default Password Expiry Duration" .
pin_fld_mask_file	cm	Specifies the location of file used to control masking of sensitive data in flists returned from the CM. See "About Securing Sensitive Customer Data with Masking" in <i>BRM Managing Customers</i> for more information.
primary_database		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 <i>BRM Installation Guide</i> .
sample_handler_logfile		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. See "Enabling In-Flight Changes to Service Orders" in <i>BRM Telco Integration</i> .

Data Manager (DM) pin.conf Entries

[Table 44–2](#) lists the DM **pin.conf** entries.

Table 44–2 *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 <i>BRM Managing Customers</i> .
crypt	NA	Associates a four-byte tag with an encryption algorithm and secret key combination. See "Configuring the Data Manager (DM) for AES Encryption" in <i>BRM Developer's Guide</i> .
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 back end to start before the DM times out. See "Setting How Long the DM Waits for the Background Startup Process to Finish" .
dm_logfile	dm	Specifies the full path to the log file used by the CM. See "Changing the Name or Location of a Log File" .
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 <i>BRM Installation Guides</i> .
dm_n_be	dm	Specifies the number of back ends the program creates and uses. See "Configuring DM Front Ends and Back Ends" .
dm_n_fe	dm	Specifies the number of front ends the program creates and uses. See "Configuring DM Front Ends and Back Ends" .
dm_n_op_fifo	dm	Specifies the number of pipelines that can be used by the back-end processes. See "Configuring Multiple Pipelines in the DM to Improve Performance" .
dm_port	dm	Specifies the port number for this DM. See "About Connecting BRM Components" .
dm_restart_children	dm	Specifies whether to replace child processes. See "Replacing Failed DM Child Processes" .

Table 44–2 (Cont.) DM pin.conf Entries

Name	Program	Description
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 <i>BRM Developer's Guide</i> .
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 'PINPAPI_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 to not try running the SQL statement if the error occurs. See "Problem: ORA-01502: Index 'PINPAPI_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" .
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 <i>BRM Developer's Guide</i> .
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 <i>BRM Developer's Guide</i> .
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 <i>BRM Developer's Guide</i> .
dd_write_enable_types	dm	Specifies whether this DM can edit, create, and delete custom object type names in the data dictionary.

Table 44–2 (Cont.) DM pin.conf Entries

Name	Program	Description
dm_xa_trans_timeout_in_secs	dm	<p>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.</p> <p>This timeout is used only if the application server XA transaction timeout is not specified.</p> <p>The minimum value is 10, and the maximum value is 5184000 (60 days). The default is 3600 (1 hour).</p> <p>See "Changing the XA Transaction Timeout Period" in <i>BRM JCA Resource Adapter</i>.</p>
extra_search	dm	<p>Specifies whether to perform an extra search count (*) on subtables for optimal memory allocation.</p> <p>See "Optimizing Memory Allocation during Database Searches".</p>
sm_database	dm	<p>Specifies the database alias name.</p> <p>See "Connecting a Data Manager to the BRM Database".</p>
sm_id	dm	<p>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.</p> <p>See "Connecting a Data Manager to the BRM Database".</p>
sm_oracle_ddl	dm	<p>Specifies whether to use Data Definition Language (DDL) when object types are updated in the data dictionary tables.</p> <p>See "Using DDL When Updating the Data Dictionary Tables" in <i>BRM Developer's Guide</i>.</p>
sm_pw	dm	<p>Specifies the password for the user specified in the sm_id entry.</p> <p>See "Configuring the Data Manager (DM) for AES Encryption" in <i>BRM Developer's Guide</i>.</p>
stmt_cache_entries	dm	<p>Specifies the maximum number of SQL statement handles to cache to improve performance.</p> <p>See "How to Use the Statement-Handle Cache".</p>

EAI Manager pin.conf Entries

[Table 44–3](#) lists the EAI Manager **pin.conf** entries.

Table 44–3 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 <i>BRM Developer's Guide</i> .
dm_http_agent	dm	Specifies a pointer to the HTTP agent. See "Configuring EAI Manager to Publish to an HTTP Port" in <i>BRM Developer's Guide</i> .
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 <i>BRM Developer's Guide</i> .
dm_http_error_opt	dm	Specifies the error handling option: <ul style="list-style-type: none"> ■ 0: Reports the error. This is the default. ■ 1: Ignores the error. See "Configuring EAI Manager to Publish to an HTTP Port" in <i>BRM Developer's Guide</i> .
dm_http_header_attrs	dm	Specifies the custom header attributes to post, in the following format: <i>attr_name1=attr_value1;attr_name2=attr_value2;</i> The maximum number of attributes allowed is 10. See "Configuring EAI Manager to Publish to an HTTP Port" in <i>BRM Developer's Guide</i> .
dm_http_header_send_host_name	dm	Specifies that the DM will send the host name as part of the header. See "Configuring EAI Manager to Publish to an HTTP Port" in <i>BRM Developer's Guide</i> .
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 <i>BRM Developer's Guide</i> .
dm_http_url	dm	(Optional) Specifies the complete URL of the HTTP server. See "Configuring EAI Manager to Publish to an HTTP Port" in <i>BRM Developer's Guide</i> .
eai_pointer	cm	Specifies where to find the EM that provides the opcode for EAI Manager. See "Configuring the EAI DM" in <i>BRM Developer's Guide</i> .
em_eai_group	cm	Specifies the member opcode in a group provided by 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 <i>BRM Developer's Guide</i> .

Table 44–3 (Cont.) EAI Manager pin.conf Entries

Name	Program	Description
em_pointer	cm	Specifies where to find EMs that provide other opcodes to the CM. See "Configuring the CM to Send Real-Time Requests to the NET_EM Module" .
enable_publish	fm_publish	Enables publishing of business events by using EAI Manager. See "Configuring the Connection Manager for EAI" in <i>BRM Developer's Guide</i> .
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 <i>BRM Developer's Guide</i> .

In-Memory Database (IMDB) Cache DM pin.conf Entries

[Table 44–4](#) lists the IMDB Cache DM **pin.conf** entries.

Table 44–4 IMDB Cache DM pin.conf Entries

Name	Program	Description
dm_n_be	dm	Specifies the number of back ends this DM creates and uses. See "Configuring DM Front Ends and Back Ends" .
dm_restart_children	dm	Specifies whether to replace child processes. See "Replacing Failed DM Child Processes" .
dm_trans_consistency_enabled	dm	Enables the transaction consistency feature. See "Enabling the Transaction Consistency Feature" .
dm_trans_timeout_in_secs	dm	Specifies how long, in seconds, a global transaction should remain in an active state after the application handling the transaction fails. Base the time on how long your system would take to recover from a failure or a node switchover. The minimum value is 10 , and the maximum value is 5184000 (60 days). The default is 3600 (one hour). See "Specifying How Long to Keep Transactions Active after an Error" .
tt_unionsearch_enabled	dm	Specifies whether the union search feature is enabled. See "Enabling Union Searches for Events" .
tt_unionsearch_poll_timeout	dm	Specifies the amount of time to wait, in milliseconds, before stopping the search procedure. See "Specifying the Timeout Value for Batch Polling Operations" .

Multithreaded Application (MTA) Framework pin.conf Entries

[Table 44–5](#) lists the MTA Framework **pin.conf** entries.

Table 44–5 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" .
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 <i>BRM Collecting Revenue Assurance Data</i> .
fetch_size	pin_mta	Specifies the number of objects received from the database in a block and cached in system memory for processing. See "Tuning the Account Cache Size for Billing Utilities (fetch_size)" .
hotlist	pin_mta	Specifies the path and file name of the hotlist file. See "Creating Configuration Files for BRM Utilities" .
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 <i>BRM Developer's Guide</i> .
login_name	nap	Specifies the user login name. See "Configuring System Passwords" in <i>BRM Developer's Guide</i> and "Login Information for Optional Service Integration Components" .
login_pw	nap	Specifies the login password. See "Configuring System Passwords" in <i>BRM Developer's Guide</i> and "Login Information for Optional Service Integration Components" .
login_type	nap	Specifies whether a login name and password are required. See "Configuring System Passwords" in <i>BRM Developer's Guide</i> and "Login Information for Optional Service Integration Components" .
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" .

Table 44–5 (Cont.) MTA Framework pin.conf Entries

Name	Program	Description
loop_forever	pin_mta	<p>Specifies whether the iterative step search for account information in the BRM database continues in a loop when the number of objects returned by a search step is 0:</p> <ul style="list-style-type: none"> ■ 0: stop the iterative step search. ■ 1: continue the iterative step search. <p>Use the loop_forever entry if the application times out before a search is complete.</p> <p>The default is 0.</p>
max_errs	pin_mta	<p>Specifies the maximum number of errors allowed in the application.</p> <p>See "Creating Configuration Files for BRM Utilities".</p>
max_time	pin_mta	<p>Specifies the maximum time, measured from application start time, for job processing before the application exits.</p> <p>See "Creating Configuration Files for BRM Utilities".</p>
multi_db	pin_mta	<p>Specifies whether the application works with a BRM multischema system.</p> <p>See "Using Multithreaded Applications with Multiple Database Schemas" in <i>BRM Developer's Guide</i>.</p>
per_batch	pin_mta	<p>Specifies the number of account objects processed by each worker thread in batch mode.</p> <p>See "Tuning the Batch Size for Billing Utilities (per_batch)".</p>
per_step	pin_mta	<p>Specifies the number of account objects returned by each search step.</p> <p>See "Setting the Batch Size for Invoice Utilities (per_step)".</p>
respawn_threads	pin_mta	<p>Specifies whether to respawn worker threads if they exit because of an error.</p> <p>See "Creating Configuration Files for BRM Utilities".</p>
retry_mta_srch	pin_mta	<p>Retries MTA searches when an insufficient memory error occurred.</p> <p>See "Creating Configuration Files for BRM Utilities".</p>
return_worker_error	pin_mta	<p>Specifies whether to return an error code when an error occurs in any of the threads.</p> <p>See "Creating Configuration Files for BRM Utilities".</p>
sleep_interval	pin_mta	<p>Specifies the time in seconds before the iterative step search for account information in the BRM database is resumed within a loop. The default is 60.</p>
userid	cm	<p>Specifies the CM for logging into the DM.</p> <p>See "Configuration Entry Syntax".</p>

business_params Reference

This chapter lists the **business_params** entries used for configuring Oracle Communications Billing and Revenue Management (BRM).

See also ["Business Logic pin.conf Reference"](#) and ["System Administration pin.conf Reference"](#) for more information.

Accounts Receivable business_params Entries

[Table 45–1](#) lists the Accounts Receivable **business_params** entries.

Table 45–1 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 <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.
ItemOverallocation	Enables the allocation of an item beyond its due date. See "Enabling Overallocation to an Item" in <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.
BalanceMonitoring	Enables balance monitoring. See "Enabling Balance Monitoring in BRM" in <i>BRM Managing Accounts Receivable</i> .	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 <i>BRM Developer's Guide</i> .	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 <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	Cached by the CM. Restart the CM after changing this entry.
PaymentSuspense	Enables payment suspense management. See "Enabling Payment Suspense in BRM" in <i>BRM Configuring and Collecting Payments</i> .	Cached by the CM. Restart the CM after changing this entry.

Table 45–1 (Cont.) Accounts Receivable business_params Entries

Name	Description	Implementation
SearchBillAmount	Enables BRM to search for a bill whose total due amount matches a specified payment amount. See "Finding Bills by Due Amount" in <i>BRM Configuring and Collecting Payments</i> .	Cached by the CM. Restart the CM after changing this entry.
SortValidityBy	Specifies the default consumption rule when consuming validity-based subbalances. See "Setting the Default Consumption Rule" in <i>BRM Setting Up Pricing and Rating</i> .	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. See "Configuring Write-Offs and Write-Off Reversals" and "How BRM Performs Write-Offs" in <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.

Activity business_params Entries

Table 45–2 lists the Activity **business_params** entries.

Table 45–2 Activity business_params Entries

Name	Description	Implementation
SetFirstUsageInSession	<p>Specifies when the start time for resource validity periods based on first usage is set.</p> <p>When this parameter is enabled, the validity period's start time is set to the start time of the first usage session, regardless of whether the session occurs or is canceled.</p> <p>When this parameter is disabled (the default), the validity period's start time is set to the end time of the first usage session.</p> <p>See "About Setting Resource Validity Periods Based on First Usage" in <i>BRM Setting Up Pricing and Rating</i>.</p>	Cached by the CM. Restart the CM after changing this entry.

Billing business_params Entries

Table 45–3 lists the Billing **business_params** entries.

Table 45–3 Billing business_params Entries

Name	Description	Implementation
AcctCycleDelayPeriod	<p>Use when billing occurred after an event was rated by Pipeline Manager 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.</p> <p>See "Configuring an Accounting Cycle Delay Period" in <i>BRM Configuring Pipeline Rating and Discounting</i>.</p>	Not cached by the CM.
AllowCorrectivePaidBills	<p>Determines whether to allow a corrective bill for a bill that is fully or partially paid.</p> <p>See "Corrective Billing" in <i>BRM Configuring and Running Billing</i>.</p>	Cached by the CM. Restart the CM after changing this entry.
ApplyCycleFeeForBillNow	<p>Indicates whether to apply cycle forward arrears and cycle arrears fees for Bill Now.</p> <p>See "Prorating Cycle Arrears and Cycle Forward Arrears for Bill Now" in <i>BRM Configuring and Running Billing</i>.</p>	Cached by the CM. Restart the CM after changing this entry.
AutoTriggeringLimit	<p>Suppresses auto-triggered billing for events processed after the billing_delay interval.</p> <p>See "Disabling Auto-Triggered Billing by Setting AutoTriggeringLimit" in <i>BRM Configuring and Running Billing</i>.</p>	Cached by the CM. Restart the CM after changing this entry.
BillingCycleOffset	<p>Specifies the hours of the day when the accounting and billing cycles start.</p> <p>See "Configuring the Billing Cutoff Time" in <i>BRM Configuring and Running Billing</i>.</p>	Cached by the CM. Restart the CM after changing this entry.

Table 45–3 (Cont.) Billing business_params Entries

Name	Description	Implementation
BillingFlowDiscount	Indicates the order of billing for discount-parents and discount-members (that is, who is billed first). See "Setting Up Billing for Sponsorship" in <i>BRM Configuring and Running Billing</i> .	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 Sponsorship" in <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Configuring Pipeline Rating and Discounting</i> .	Cached by the CM. Restart the CM after changing this entry.
CacheResidencyDistinction	Enables BRM to handle all objects as postpaid. Cache residency objects are objects associated with prepaid subscribers. See "Enabling Cache Residency Distinction" and "About Convergent BRM Systems" .	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 Threshold Amount for Corrective Bills" in <i>BRM Configuring and Running Billing</i> .	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. See "Creating Two Bills during the Delayed Billing Period" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
DefaultBusinessProfile	Sets the default cache type for the system. By default, all reference object instances get assigned an object cache type value of DEFAULT (Convergent business profile) if they have not been associated with a business profile. See "Changing the Default Business Profile" .	Cached by the CM. Restart the CM after changing this entry.
EnableARA	Enables Revenue Assurance Manager. See "Configuring Bill Now, On-Demand Billing, and Auto-Triggered Billing to Collect Revenue Assurance Data" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
EnableCorrectiveInvoices	Enables corrective billing and corrective invoicing. See "Enabling Corrective Billing in BRM" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
EventChargeDiscountMode	Defines the mode for the PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT opcode to retrieve event details.	Cached by the CM. Restart the CM after changing this entry.

Table 45–3 (Cont.) Billing business_params Entries

Name	Description	Implementation
GenerateCorrectiveBillNo	Determines the prefix and sequence number to assign to the corrective bill. See "Setting Up Bill Numbers for Corrective Bills" in <i>BRM Configuring and Running Billing</i> .	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 <i>BRM Setting Up Pricing and Rating</i> .	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 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 <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
NonCurrencyResourceJournaling	Controls the creation of /journal objects for noncurrency resources. See "Disabling /journal Objects for Non-Currency Resources" in <i>BRM Collecting General Ledger Data</i> .	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 <i>BRM Configuring and Collecting Payments</i> .	Cached by the CM. Restart the CM after changing this entry.

Table 45–3 (Cont.) Billing business_params Entries

Name	Description	Implementation
PerfAdvancedTuningSettings	<p>Improves billing performance.</p> <p>The following values can be set for the PerfAdvancedTuningSettings business parameter.</p> <p>0 (No Bit is set): Billing will be processed with default settings.</p> <p>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.</p> <p>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.</p> <p>4 (0x04 Bit 2 is set): Used internally by BRM</p> <p>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 Previous Total Unpaid Bill for Open Item Accounting Type".</p> <p>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.</p> <p>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.</p> <p>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 Trial Billing Performance by Enabling General Ledger Collection".</p> <p>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).</p> <p>Important: Do not set this business parameter to any value other than the valid values or any combination of those.</p>	<p>Cached by the CM. Restart the CM after changing this entry.</p>

Table 45–3 (Cont.) Billing business_params Entries

Name	Description	Implementation
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 <i>BRM Managing Customers</i> .	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. See "Processing Payments for Previous Bills" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
RemoveSponsoree	Determines whether to remove a member account from sponsor groups. See "How Account Status Changes Affect Sponsor Groups" in <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.
RerateDuringBilling	Enables borrowing from the rollover amount during the current billing cycle. See "Modifying Business Parameters to Enable Rerating and Rollover Correction" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
RolloverCorrectionDuringBilling	Enables borrowing from the rollover amount during the current billing cycle. See "Modifying Business Parameters to Enable Rerating and Rollover Correction" in <i>BRM Configuring and Running Billing</i> .	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 with other discounts that are valid during the same period. See "Enabling Sequential Discounting of Cycle Fees" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	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 subbalance) to use first. You use resource consumption rules to specify the order in which resource subbalances are consumed, according to the validity start time and end time. See "Setting the Default Consumption Rule" in <i>BRM Setting Up Pricing and Rating</i> .	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 "Splitting Sponsored Charges into Multiple Items" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
SubBalValidity	Enables you to extend the validity period of the original subbalance when a subscription service is transferred. See "Configuring Subbalance Validity for Subscription Service Transfer" in <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.

Table 45–3 (Cont.) Billing business_params Entries

Name	Description	Implementation
ValidateDiscountDependency	Enables discount exclusion rules, which establish a mutually exclusive relationship between discounts or between a discount and a plan. See "Configuring and Defining Exclusion Rules" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Cached by the CM. Restart the CM after changing this entry.

Customer Management business_params Entries

Table 45–4 lists the Customer Management **business_params** entries.

Table 45–4 Customer Management business_params Entries

Name	Description	Implementation
AutomatedMonitorSetup	Enables automated balance monitoring. See "Enabling AMS in BRM" in <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.
BestPricing	Enables the best pricing feature. If best pricing is enabled, the best pricing calculation is performed for the member service at billing. See "Enabling Best Pricing" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
CancelFullDiscountImmediate	Enables BRM to cancel discounts immediately when the discount validity rule is set to Full Discount . See "Managing Discount End Dates During Mid-Cycle Cancellations" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Cached by the CM. Restart the CM after changing this entry.
DiscountBasedOnContractDaysFeature	Enables the contract days counter (CDC), which provides a discount based on the cumulative number of contract days. See "Enabling Support for Discounts Based on Contract Days" in <i>BRM Configuring Pipeline Rating and Discounting</i> .	Cached by the CM. Restart the CM after changing this entry.
EnablePasswordRestriction	Enables passwords to secure the creation, modification, and deletion of /service objects. See "Enabling Password Restriction for /service Objects" in <i>BRM Setting Up Pricing and Rating</i> .	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 finish without requiring immediate intervention to fix the errors. See "Enabling Payment Suspense in BRM" in <i>BRM Configuring and Collecting Payments</i> .	Cached by the CM. Restart the CM after changing this entry.

Table 45–4 (Cont.) Customer Management business_params Entries

Name	Description	Implementation
RolloverTransfer	Enables rollover transfers. See "Enabling Rollover Transfers in BRM" in <i>BRM Managing Accounts Receivable</i> .	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 <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.
SubsDis74BackDateValidations	Enables users to create accounts with services or resources backdated before the account creation date. See "Allowing Accounts To Be Created with Backdated Services or Resources" in <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.

General Ledger business_params Entries

[Table 45–5](#) lists the General Ledger **business_params** entries.

Table 45–5 General Ledger business_params Entries

Name	Description	Implementation
CustomJournalUpdate	Enables custom updates to general ledger data. See "Enabling Custom Updates to G/L Data" in <i>BRM Collecting General Ledger Data</i> .	Cached by the CM. Restart the CM after changing this entry.
GeneralLedgerReporting	Use general ledger reporting. See "Enabling and Disabling General Ledger Collection in BRM" in <i>BRM Collecting General Ledger Data</i> .	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.	Cached by the CM. Restart the CM after changing this entry.

Invoicing business_params Entries

[Table 45–6](#) lists the Invoicing **business_params** entries.

Table 45–6 Invoicing business_params Entries

Name	Description	Implementation
ADSTTaxHandle	Groups taxes on invoices based on the tax supplier IDs. See "Aggregating Taxes on Invoices" in <i>BRM Designing and Generating Invoices</i> .	Cached by the CM. Restart the CM after changing this entry.
EnableInvoicingIntegration	Generates invoice documents by using the BRM-BI Publisher integration framework. See the discussion on enabling the BRM-BI Publisher integration in <i>BRM Designing and Generating Invoices</i> .	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 the discussion on specifying the default format in which to store invoices in <i>BRM Designing and Generating Invoices</i> .	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 <i>BRM Designing and Generating Invoices</i> .	Cached by the CM. Restart the CM after changing this entry.
SubARItemsIncluded	Displays A/R items for each subordinate account on the parent invoice. See "Setting Defaults for Hierarchical Group Invoices" in <i>BRM Designing and Generating Invoices</i> .	This entry is read by the utility when it runs. You do not need to restart the CM.
ThresholdSubordsDetail	Sets the threshold for including subordinate account details on parent detailed invoices. See "Setting Defaults for Hierarchical Group Invoices" in <i>BRM Designing and Generating Invoices</i> .	This entry is read by the utility when it runs. You do not need to restart the CM.
ThresholdSubordsSummary	Sets the threshold for including subordinate account details on parent summary invoices. See "Setting Defaults for Hierarchical Group Invoices" in <i>BRM Designing and Generating Invoices</i> .	This entry is read by the utility when it runs. You do not need to restart the CM.

Pricing and Rating business_params Entries

[Table 45–7](#) lists the Pricing and Rating **business_params** entries.

Table 45–7 Pricing and Rating business_params Entries

Name	Description	Implementation
AllocateReratingAdjustments	Determines whether to allocate automatic adjustments from rerating to original bills See "Corrective Billing" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
ApplyDeferredTaxDuringRerating	Enables deferred tax calculation during rerating. See "Enabling Calculation of Deferred Taxes During Rerating" in <i>BRM Setting Up Pricing and Rating</i> .	Cached by the CM. Restart the CM after changing this entry.
BatchRatingPipeline	Enables Account Synchronization to pass business events between pin_rerate and Pipeline Manager. See "Specifying Whether the Batch Rating Pipeline Is Enabled" in <i>BRM Setting Up Pricing and Rating</i> .	Cached by the CM. Restart the CM after changing this entry.
ECERating	Enables Oracle Communications Elastic Charging Engine (ECE) rating. See "Enabling Elastic Charging Engine Rerating" in <i>BRM Setting Up Pricing and Rating</i> .	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 <i>BRM Managing Accounts Receivable</i> .	Cached by the CM. Restart the CM after changing this entry.
EnableTailormadeCache	Specifies if tailor-made products must be maintained in the rating cache. See " Enabling and Disabling the Caching of Customized Products ".	Cached by the CM. Restart the CM after changing this entry.
OfferEligibilitySelectionMode	Enables BRM to rerate events by using the rates applied when rating conditions change during the session. See "Enabling Rerating when the Rating Conditions Change During the Session" in <i>BRM Setting Up Pricing and Rating</i> .	Cached by the CM. Restart the CM after changing this entry.
OverrideCreditLimit	Enables system-wide credit limit override. See "Configuring the System-Wide Override Credit Limit" in <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.

Table 45–7 (Cont.) Pricing and Rating business_params Entries

Name	Description	Implementation
ProductsDiscountsThreshold	Specifies the maximum number of products or discounts that can be cached by the real-time rating engine. See "Configuring the Maximum Number of Products and Discounts Cached" .	Cached by the CM. Restart the CM after changing this entry.
RestrictResourceValidityToOffer	Restricts resource validity end time to the end time of the product or discount that grants the resource. See "Configuring Real-Time Rating to Restrict Resource Validity End Time" in <i>BRM Setting Up Pricing and Rating</i> .	Cached by the CM. Restart the CM after changing this entry.
TimestampRoundingForPurchase Grant	Enables time-stamp rounding to midnight for the resources granted by purchase events. See "Configuring Time-Stamp Rounding for the Purchase Grants" in <i>BRM Setting Up Pricing and Rating</i> .	Cached by the CM. Restart the CM after changing this entry.

Selective Account Loading business_params Entries

[Table 45–8](#) lists the Selective Account Loading **business_params** entries.

Table 45–8 Selective Account Loading business_params Entry

Name	Description	Implementation
CacheResidenciesForBatchPipeline	Enables Pipeline Manager to selectively load accounts. See "Configuring Pipeline Manager for Selective Account Loading" .	Cached by the CM and Pipeline Manager. Restart the CM and Pipeline Manager after changing this entry.

Subscription business_params Entries

[Table 45–9](#) lists the Subscription **business_params** entries.

Table 45–9 Subscription business_params Entries

Name	Description	Implementation
AllowBackdateNoRerate	Enables backdating beyond the number of billing cycles specified in the num_billing_cycles entry without requesting to automatically rerate. See "Backdating beyond Configured Billing Cycles without Automatic Rerating Request" in <i>BRM Setting Up Pricing and Rating</i> .	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 product whose validity expires within the first cycle after the product 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 the discussion about using two events to prorate charges for products whose validity ends in first cycle in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
EventAdjustmentsDuringCancellation	Specifies whether to include event adjustments when calculating product cancellation refunds. See "Including Event Adjustments in Product Cancellation Refunds" in <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.
TransferScheduledActions	Specifies whether to transfer pending scheduled actions associated with an existing subscription when transferring the subscription to a different account. See "Transferring a Subscription Service" in <i>BRM Managing Customers</i> .	Cached by the CM. Restart the CM after changing this entry.
UsePrioritySubscriptionFees	Specifies whether to use product priority while applying cycle fees for products. See "Enabling Product Priority While Applying Cycle Fees" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.
ApplyProrationRules	Allows you to apply proration rules for purchases or cancellations. See "Enabling BRM to Apply Proration Rules" in <i>BRM Configuring and Running Billing</i> .	Cached by the CM. Restart the CM after changing this entry.

System Administration business_params Entries

Table 45–10 lists the System Administration **business_params** entries.

Table 45–10 System Administration business_params Entries

Name	Description	Implementation
MaxLoginAttempts	Specifies the maximum number of invalid login attempts to be 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.

