Oracle® Communications Billing and Revenue Management

Opcode Guide

Release 15.1
F93203-01
April 2025

ORACLE

Oracle Communications Billing and Revenue Management Opcode Guide, Release 15.1

F93203-01

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

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

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

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

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

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

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

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

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

Contents

Preface

Audience xxix
Documentation Accessibility xxix
Diversity and Inclusion xxix

Part I Opcode Reference

1 Opcode Descriptions

About Using Opcodes 1-1
- About the Opcode Visibility Value 1-1
- About the Opcode Transaction Value 1-2

Account Dump FM Policy Opcodes 1-2
Account Dump FM Standard Opcode 1-3
Accounts Receivable FM Policy Opcodes 1-3
Accounts Receivable FM Standard Opcodes 1-3
Account Synchronization FM Opcodes 1-6
Activity FM Policy Opcodes 1-7
Activity FM Standard Opcodes 1-8
Balance FM Policy Opcodes 1-9
Balance FM Standard Opcodes 1-10
Base Opcodes 1-11
Batch Suspense Manager FM Standard Opcodes 1-14
Billing FM Policy Opcodes 1-14
Billing FM Standard Opcodes 1-16
Channel FM Standard Opcodes 1-19
Collections Manager FM Policy Opcodes 1-19
Collections Manager FM Standard Opcodes 1-21
Contract FM Policy Opcodes 1-24
Contract FM Standard Opcodes 1-25
Customer FM Policy Opcodes 1-25
Customer FM Standard Opcodes 1-30
Customer Care FM Standard Opcodes 1-35

Deposit FM Policy Opcodes 1-35
Deposit FM Standard Opcodes 1-37
Device FM Policy Opcodes 1-39
Device FM Standard Opcodes 1-40
Email Data Manager Opcodes 1-40
General Ledger FM Policy Opcodes 1-41
General Ledger FM Standard Opcode 1-41
GPRS Manager 3.0 FM Policy Opcode 1-42
GPRS Manager 3.0 FM Standard Opcode 1-42
Group FM Standard Opcodes 1-42
GSM Manager FM Policy Opcode 1-43
GSM Manager FM Standard Opcode 1-43
Installment FM Policy Opcodes 1-44
Installment FM Standard Opcodes 1-44
Invoicing FM Policy Opcodes 1-45
Invoicing FM Standard Opcodes 1-46
IP Address Manager APN FM Policy Opcodes 1-47
IP Address Manager FM Policy Opcodes 1-48
IP Address Manager FM Standard Opcodes 1-48
Job FM Policy Opcodes 1-49
Job FM Standard Opcodes 1-49
LDAP Base Opcodes 1-50
Loan FM Policy Opcodes 1-51
Loan FM Standard Opcodes 1-52
Monitor FM Standard Opcodes 1-52
Notification FM Standard Opcodes 1-54
Notification FM Policy Opcodes 1-54
Number Manager FM Policy Opcodes 1-55
Number Manager FM Standard Opcodes 1-56
Offer Profile FM Standard Opcode 1-56
Order FM Policy Opcodes 1-57
Order FM Standard Opcodes 1-57
Payment FM Policy Opcodes 1-58
Payment FM Standard Opcodes 1-60
Permissioning FM Standard Opcodes 1-63
Process Audit FM Policy Opcodes 1-64
Process Audit FM Standard Opcodes 1-64
Provisioning FM Policy Opcode 1-65
Provisioning FM Standard Opcodes 1-65
Publish FM Policy Opcodes 1-66
Rating FM Policy Opcodes 1-66
Rating FM Standard Opcodes 1-67

Remittance FM Policy Opcodes 1-68
Remittance FM Standard Opcodes 1-68
Replication FM Policy Opcode 1-69
Rerating FM Standard Opcode 1-69
SDK FM Standard Opcodes 1-69
Services Framework Manager FM Policy Opcodes 1-70
Services Framework Manager FM Provisioning Opcodes 1-71
SIM Manager FM Policy Opcodes 1-72
SIM Manager FM Standard Opcodes 1-73
Subscription Management FM Policy Opcodes 1-73
Subscription Management FM Standard Opcodes 1-75
Suspense Manager FM Standard Opcodes 1-80
Universal Message Store (UMS) FM Standard Opcodes 1-81

2 Context Management Opcodes

Context Management Opcodes 2-1
PCM_CONNECT 2-2
PCM_CONTEXT_CLOSE 2-3
PCM_CONTEXT_OPEN 2-4
PCM_OP 2-7
PCM_OPREF 2-9

Part II Opcode Workflows

3 About Opcode Workflows

4 Accounts Receivable Opcode Workflows

Opcodes Described in This Chapter 4-1
Getting a List of A/R Items 4-5
Reversing Refunds 4-6
Transferring Amounts between Items 4-7
Transferring Balances between Items 4-7
Transferring from a Negative to a Positive Balance 4-9
Transferring Balances between Balance Groups 4-9
Customizing Item Transfer Validation 4-10
Managing Currency and Non-Currency Balance Transfers between Accounts or Services 4-10
Transferring Balances between Accounts or Services 4-11
Customizing Balance Transfers 4-12
Transferring Services between Balance Groups 4-12

Performing Adjustments 4-16
Adjusting Accounts, Subscription Services, and Member Services 4-16
Fields You Should Include in the Flist for Account, Subscription Service, and Member Service Adjustments 4-17
Flags for Account, Subscription Service, and Member Service Adjustments 4-18
Adjusting Bills 4-18
Fields You Should Include in the Flist for Bill Adjustments 4-19
Flags for Bill Adjustments 4-20
Adjusting Items 4-21
Fields You Should Include in the Input Flist for Item Adjustments 4-23
Flags for Item Adjustments 4-23
Customizing Item Adjustments 4-23
Adjusting Events 4-24
Fields You Should Include in the Input Flist for Event Adjustments 4-25
Flags for Event Adjustments 4-25
Categorizing Unbilled Event Adjustments in G/L Reports 4-26
Including Taxes in the Adjustment 4-27
Tax Processing for Account Adjustments 4-27
Tax Processing for Bill and Item Level Adjustments 4-27
Tax Processing for Event Adjustments 4-28
Including Reason Codes in the Adjustment 4-28
Assigning G/L IDs for an Adjustment 4-29
Applying Debits and Credits 4-29
Flags for Applying Debits and Credits 4-30
Performing Disputes and Settlements 4-30
Disputing Bills 4-31
Fields You Should Include in the Input Flist for Bill Disputes 4-32
Flags for Bill Disputes 4-33
Settling Disputed Bills 4-33
Fields You Should Include in the Input Flist 4-34
Flags for Bill Settlements 4-34
Disputing Items 4-35
Fields You Should Include in the Input Flist 4-36
Flags for Item Disputes 4-37
Customizing Item Disputes 4-37
Settling Disputed Items 4-38
Fields You Should Include in the Input Flist for Item Settlements 4-39
Flags for Item Settlements 4-40
Customizing Item Settlements 4-40
Disputing Events 4-40
Fields You Should Include in the Input Flist for Event Disputes 4-41
Flags for Event Disputes 4-42

Settling Disputed Events 4-42
Fields You Should Include in the Input Flist for Event Settlements 4-44
Flags for Event Settlements 4-44
Configuring the Tax Treatment for Adjustments, Disputes, and Settlements 4-45
Including Taxes in the Dispute or Settlement 4-45
Tax Processing for Disputes 4-45
Tax Processing for Settlements 4-46
Including Reason Codes in the Dispute or Settlement 4-47
Assigning G/L IDs for a Dispute or Settlement 4-47
Writing Off Debts and Reversing Write-Offs 4-47
About Initiating Write-Offs 4-47
Performing Write Offs 4-48
About Account Write-Offs 4-49
About Bill Unit Write-Offs 4-50
About Bill Write-Offs 4-50
Flags You Should Use for Write-Offs 4-50
About Taxes for Write-Offs 4-51
Customizing Write-Off Validation 4-51
About Initiating Write-Off Reversals 4-51
Reversing Write-Offs 4-52
Reversing a Write-Off Reversal 4-53
Customizing Write-Off Reversals 4-53
Customizing the Rules for Performing Write-Off Reversals 4-53
About Automatic Write-Off Reversals During Payment Collection 4-54
Customizing Reversal of Payments Allocated to Written-Off Accounts 4-54
Retrieving A/R Information 4-56
Finding a Bill 4-56
Finding a Bill Unit 4-57
Finding a Balance Group and Its Balances 4-57
Finding a Balance Group and Service for Bill Units 4-58
Getting a List of Bills 4-59
Getting Bill Items 4-60
Getting Bills 4-60
Finding Items 4-61
Finding Discounts in Bill Items 4-62
Retrieving a Balance Summary 4-62
Retrieving a List of Bills for a Bill Unit 4-63
About the Bill Data Retrieved 4-63
Specifying the Search Criteria for Retrieving Bills 4-64
Retrieving A/R Items That Apply to a Bill Unit 4-65
About the Item Data Retrieved 4-65
Specifying Search Criteria for Retrieving Items 4-66

Retrieving a List of Bill Items for a Bill Unit 4-67
About the Bill Item Data Retrieved 4-68
Customizing the Number of Items to Retrieve in a Search 4-68
Specifying Search Criteria for Bill Items 4-68
Retrieving Details about a Specific A/R Item or Bill Item 4-69
About the Item Data Retrieved 4-69
About the Item Detail Data Retrieved 4-70
Specifying Search Criteria for Retrieving Items 4-71
About the Item Tax Detail Retrieved 4-72
Retrieving Dispute Details for a Bill Unit 4-72
Retrieving a Full Set of Dispute Data 4-73
Data Retrieved by PCM_OP_AR_GET_DISPUTES 4-73
Specifying Search Criteria for PCM_OP_AR_GET_DISPUTE_DETAILS 4-73
Returning a Limited Set of Dispute Data 4-74
Data Retrieved by PCM_OP_AR_GET_DISPUTES 4-74
Specifying Search Criteria for PCM_OP_AR_GET_DISPUTES 4-74
Retrieving Details on Available Balances 4-75
Finding Events Associated with an Account 4-75
Finding Events Associated with Bill Items 4-76
Performing Rollover Transfers 4-76

5 Activity Opcode Workflows

Opcodes Described in This Chapter 5-1
Managing Sessions 5-2
Starting Sessions 5-2
Recording the Start of a Session 5-3
Updating a Session Event 5-3
Recording the End of a Session 5-4
Recording Session Events 5-4
Loading Sessions 5-4
Customizing News Feed Values and Events 5-4
Managing Operation Failure Records 5-5
Recording Failed Operations 5-6
Retrieving Operation Failure Records 5-6

6 Billing Opcode Workflows

Opcodes Described in This Chapter 6-1
Creating Bills 6-4
Modifying a Bill Object 6-5
Calculating When a Bill is Due 6-5

Customizing Bill Due Date Calculations for Payment Terms 6-6
Customizing the Format of Bill and Invoice Numbers 6-9
Running Bill Now 6-10
Customizing Bill Now 6-11
Running Bill Now for a Service 6-11
Applying Discounts and Folds with Bill Now 6-11
Changing the Bill Now Due Date 6-12
On-Purchase Billing 6-12
Creating /item/sponsor Objects 6-12
Customizing Billing 6-13
Customizing the Minimum Amount to Charge 6-13
Customizing Accounting Cycles 6-14
Customizing How to Bill Events That Occur between Billing Cycles 6-14
Customizing Credit Limit and Consumption Rules 6-15
Customizing Client Applications to Modify Fixed Thresholds 6-15
About Currency Conversion 6-15
Managing Bill Units 6-16
Creating Bill Units 6-16
Setting the Billing Day of Month 6-17
Updating Bill Units 6-17
Preparing Bill Unit Data 6-18
How BRM Calculates Long Billing Cycles 6-19
Setting the Billing DOM According to the Payment Method 6-21
Assigning Default Billing Information 6-21
Determining the Accounting Type 6-21
Assigning Billing DOMs to Bill Units 6-22
Assigning Billing DOMs Based on the Billing Segment 6-22
Customizing the DOM Assignment Process 6-23
Validating Bill Unit Data 6-24
Validating Billing Segment Information 6-25
Suspending and Resuming Billing of Closed Accounts 6-25
Suspending Billing of Closed Accounts 6-25
Resuming Billing When Closed Accounts Are Reactivated 6-26
Deleting Bill Units 6-27
Setting Up Hierarchical and Sharing Relationships among Bill Units 6-27
Setting Up Hierarchical Relationships among Bill Units 6-27
Setting Up Sharing Relationships among Bill Units 6-28
Changing Hierarchical and Sharing Relationships among Bill Units 6-28
Billing Delays for Moved Bill Units 6-28
Suppressing Bills 6-28
Determining Whether Bills Should Be Suppressed 6-29
Manually Suppressing Bills 6-31

Suppressing Accounts 6-32
Ending Manual Account Suppression 6-34
Adding Bill Suppression Exceptions 6-34
Deleting Bill Suppression Exceptions 6-35
Managing Proforma Invoices 6-36
Moving Bill Items 6-38
Making Trial Bills 6-38
Collecting Split Revenue Assurance Data 6-39
Corrective Billing 6-40
Making a Corrective Bill 6-40
Validating Bills for the Corrective Billing Process 6-41
Standard Bill Validations 6-42
Policy Bill Validations 6-42
Payment Due Dates for Corrective Bills 6-43
Using Revenue Assurance Manager 6-43
Customizing Revenue Assurance Manager 6-43
Customizing Audit Object Validation 6-44
Customizing Alert Behavior 6-44
Customizing the Revenue Assurance Written-Off Event Record Summaries 6-44

7 Charging Opcode Workflows

Opcodes Described in This Chapter 7-1
Subscription Rating Opcodes 7-4
Implementing Custom Rating 7-4
Getting ERAs for an Event 7-5
Modifying Rated Events 7-5
Modifying ERAs 7-6
Managing Credit Limits and Sub-Balance Consumption Rules 7-6
How BRM Handles Consumption Rules and Credit Limits 7-7
Retrieving Credit Limits 7-8
Maintaining Subscriber's Charging Preferences Data 7-9
Modifying Events before Sending Them to ECE 7-9
Processing BRM Events That Make Up Account Synchronization Business Events 7-10
Modifying BRM Events That Make Up Account Synchronization Business Events 7-10
Rerating Opcodes 7-11
How BRM Tracks Rerated Events 7-11
How BRM Creates Rerate Jobs 7-12
How BRM Handles Duplicate Rerate Jobs 7-14
Rerating Cycle Forward and Cycle Forward Arrears Events 7-15
Customizing Event Searches for Rerating 7-16
Customizing Automatic Creation of Rerate Jobs 7-17

Creating a Custom Opcode for Rerating 7-18
Specifying a Rerate Reason Code 7-19
Specifying Selection Criteria 7-19
Specifying Price Overrides 7-20
Configuring Event Notification for Customized Automatic Rerating 7-21
Configuring Event Notification for Override Pricing 7-21
Suspense Management 7-22
Suspended Call Record States 7-22
Recycling Suspended Records 7-23
Searching for Records in a CDR File 7-23
Searching for Records with a Recycle Key 7-23
Initiating Suspense Recycling 7-23
Resubmitting Suspended Batches 7-24
Changing the Contents of Fields in Suspended Event Records 7-25
Undoing Edits to Suspended Event Records 7-26
Deleting Suspended Records 7-26
Deleting Records for Suspended Batches 7-27
Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off 7-27
Deleting Records in a CDR File 7-28
Writing Off Suspended Records 7-28
Writing Off Suspended Batches 7-29
Processing Suspended Records in Bulk 7-29
Processing Suspended Records in Multiple Steps 7-29
Editing Suspended Records in Bulk 7-30
Writing Off Suspended Records in Bulk 7-31
Deleting Suspended Records in Bulk 7-32
Provisioning Process Opcode Flow 7-33
About Creating Service Orders for Profile Changes 7-35
GPRS Opcodes 7-35
About Associating APNs and QoS with GPRS Services 7-35
Associating APNs and QoS with GPRS Services 7-36
Mapping Service Types to Service-Specific Opcodes 7-36
Associating APN and QoS Pairs with GPRS Services 7-37
Updating Custom GPRS Service Fields 7-37
Updating Custom GSM Service Fields 7-37

8 Collections Opcode Workflows

Opcodes Described in This Chapter 8-1
Customizing Collections Manager 8-4
Configuring the Currency Used in Collections 8-4

Customizing Due Dates 8-4
Customizing Dunning Letters 8-4
Applying Finance Charges 8-5
Applying Late Fees 8-6
Assigning Bill Units Automatically 8-6
Assigning Bill Units to a Debt Collections Agency 8-7
Customizing How to Apply Collections Actions to Collections Group Members 8-7
Mapping Bill Units to Collections Profiles 8-7
Performing Custom Collections Actions 8-8
Performing Custom Actions when a Bill Unit Leaves Collections 8-8
Managing Overdue Balance Collection 8-8
Creating or Updating Collections Actions 8-8
Creating or Updating Collections Profiles 8-9
Getting All Currently Defined Collections Actions 8-9
Getting All Currently Defined Collections Profiles 8-10
Getting All Currently Defined Collections Scenarios 8-10
Getting Details of a Collections Scenario 8-10
Getting a List of Message Templates 8-11
Deleting an Existing Collections Action 8-11
Deleting an Existing Collections Profile 8-11
Deleting an Existing Collections Scenario 8-12
Managing Promise-to-Pay Agreements 8-12
Creating a System-Configured Promise-to-Pay Agreement 8-13
Creating a Manually Configured Promise-to-Pay Agreement 8-14
Updating Amount and Payment Details 8-15
Canceling a Promise-to-Pay Agreement 8-15
Breaking a Promise-to-Pay Agreement 8-16
Rescheduling a Promise-to-Pay Action 8-16
Changing the Status of a Promise-to-Pay Action 8-16
Completing Promise-to-Pay Installments during Payment Processing 8-16
Managing Collections Groups 8-17
Creating a Collections Group 8-17
Adding Members to a Collections Group 8-18
Removing a Member from a Collections Group 8-18
Deleting a Collections Group 8-18
Retrieving a Collections Group 8-18
Modifying a Collections Group 8-19
Changing the Parent of a Collections Group 8-19
Performing Manual Collections Actions 8-19
Assigning Bill Units to a Collections Agent 8-20
Adding Actions to a Collections Scenario 8-20
Exempting Bill Units from Collections 8-21

Rescheduling an Action Scheduled for a Bill Unit 8-22
Changing the Status of a Collections Action 8-22
Replacing a Collections Scenario 8-23
Performing System Collections Actions 8-24
Executing Automatic Collections Actions 8-24
Preparing Invoice Reminders 8-25
Gathering and Storing Data for Dunning Letters 8-26
Retrieving Dunning Letters 8-26
Executing Pending Actions for a Bill Unit 8-26
Retrieving Collections Information 8-27
Retrieving a List of Bill Units in Collections 8-28
Retrieving Aging Buckets Information 8-28
Retrieving Scenario Information 8-29
Getting All Valid Collections Scenarios 8-29
Retrieving a List of Collections Actions 8-30
Retrieving Collections Action History Information 8-30

9 Customer Management Opcode Workflows

Opcodes Described in This Chapter 9-1
About Account Locking 9-8
Transaction Handling During Account Creation 9-8
Creating Accounts 9-9
Customizing Account Numbers 9-12
Customizing the Introductory Message 9-12
Sending the Introductory Message 9-13
Customizing Account Creation Charges 9-14
Creating Multiple Accounts in One Transaction 9-14
Creating Accounts with Backdated Services or Balances 9-14
Creating Customer's Collections Profiles at the Time of Customer Account Creation 9-15
Customizing Automatic Account Creation (AAC) Information 9-15
Adding Custom Account Creation Steps Before the Account Is Committed 9-16
Adding Custom Account Creation Steps After the Account Is Committed 9-16
Integrating BRM With an External Account Creation Application 9-16
Sending Account Information to Your Application when an Account is Created 9-17
Sending Account Information to Your Application when an Account is Modified 9-17
Returning a Point-of-Presence (POP) List 9-17
Creating Accounts in a Multischema System 9-18
Getting a List of Database Schemas 9-19
Selecting a Database Schema 9-19
Managing Name and Address Information 9-19
Customizing Name and Address Information 9-20

Customizing Locale Information 9-20
Creating Custom Country Aliases 9-21
Authenticating and Authorizing Customers 9-21
Customizing Login Names 9-21
Requiring Login Names for Email and Broadband Services 9-22
Creating Logins for Prepaid Services 9-22
Creating Logins for Email Services 9-22
Creating Passwords 9-23
Customizing Passwords 9-23
Implementing Password Encryption 9-23
Creating Passwords for Prepaid Services 9-24
Customizing Password Expiration 9-24
Authenticating Customers 9-24
Authenticating User Actions 9-24
Customizing Authentication Checks 9-25
Enabling Duplicate Session Checking 9-26
Managing and Customizing Profiles 9-27
Retrieving Account Profile Information 9-28
Modifying an Account 9-28
Changing a Bill's Payment Method 9-30
Creating Services 9-31
Modifying Services 9-32
Finding Accounts 9-33
Finding a Customer's Account Information 9-33
Deleting Accounts 9-34
Customizing Customer Payment Information 9-34
Customizing Payment Method Data Preparation 9-36
Specifying the Payment Processor Vendor 9-36
Customizing Payment Method Validation 9-36
Default /payinfo Validation 9-37
CVV2/CID Fraud Prevention Functionality 9-37
Verifying the Maximum Number of CVV2 Digits 9-38
Disabling the Credit Card Checksum 9-38
Customizing the Banking Information for US and Canadian Direct Debit 9-39
Customizing the Account Used for Credit Card Validation 9-39
Managing Mandate Information 9-39
Registering a Mandate 9-39
Updating a Mandate 9-40
Canceling a Mandate 9-40
Setting Account, Service, and Bill Unit Status 9-41
Changing the Status of an Account, Bill Unit, or Service 9-41
PIN_FLD_STATUS Field Values for PCM_OP_CUST_SET_STATUS 9-41

PIN_FLD_STATUS_FLAG Field Values for PCM_OP_CUST_SET_STATUS 9-42
About Deferred Actions When Using PCM_OP_CUST_SET_STATUS 9-43
Setting, Resetting, or Unsetting a Deferred CLOSE 9-43
Customizing Status Changes 9-44
Inactivating Accounts that Exceed a Specified Limit 9-44
Backdating Status Changes 9-44
Getting Life Cycle States 9-44
Amending Creditor Information 9-45
Amending a Mandate 9-45
Canceling a Mandate 9-46
Managing Deferred Actions 9-46
Scheduling Deferred Actions 9-46
Modifying Deferred Actions 9-47
Deleting Deferred Actions 9-47
Executing Deferred Actions 9-47
Performing Policy Checks before Scheduling Deferred Actions 9-47
Managing Service Groups 9-47
Creating a Service Group 9-48
Configuring Extended Rating Attributes for a Service Group 9-48
Associating a Device with a Service Group 9-49
Adding a Service to an Existing Service Group 9-49
Canceling a Service Group 9-49
Transferring Service Groups between Accounts in the Same Schema 9-50
Enabling Transfer of Pending Scheduled Actions during Service Group Transfers 9-52
Transferring Service Groups between Accounts in Different Schemas 9-53
Enabling Transfer of Service Groups between Accounts in Multiple Schemas 9-53
Transferring Service Groups across Schemas 9-53
Managing Profile Sharing Groups 9-55
Creating a Profile Sharing Group 9-55
Validating Profile Sharing Group Members 9-56
Modifying a Profile Sharing Group 9-57
Adding Members to a Profile Sharing Group 9-57
Adding Profiles to a Profile Sharing Group 9-59
Deleting Members and Profiles from a Profile Sharing Group 9-59
Changing the Owner of a Profile Sharing Group 9-60
Deleting a Profile Sharing Group 9-61
Using Access Control Lists 9-61
Managing ACL Groups 9-61
Accessing /group/acl Data 9-62
Finding CSR Membership 9-62
Customizing the Account Dump Utility (ADU) 9-62
Business Profile Opcode Workflows 9-63

Creating Business Profiles 9-63
Assigning Bill Units to Business Profiles 9-63
Changing a Bill Unit's Business Profile 9-64
Getting Information about a Business Profile 9-65
Creating and Managing Account Hierarchies 9-66
Creating Account Hierarchies 9-67
Adding Accounts to Account Hierarchies 9-67
Moving Accounts from One Account Hierarchy to Another 9-68
Deleting Accounts from Account Hierarchies 9-69
Deleting Account Hierarchies 9-69
Finding the Parent of an Account Hierarchy 9-69
Getting a List of Child Accounts in an Account Hierarchy 9-70
About the PCM_OP_GROUP Opcodes 9-70
Creating Groups 9-70
Adding Members to Groups 9-70
Deleting Members from Groups 9-71
Setting a Group Parent 9-71
Deleting Groups 9-71
Updating Inheritance Fields in Groups 9-71

10 Deposit Opcode Workflows

Opcodes Described in This Chapter 10-2
Creating Deposit Specification Profiles 10-3
Validating Deposit Specification Profiles 10-4
Getting Deposit Specification Profiles 10-4
Modifying Deposit Specification Profiles 10-4
Creating Deposit Specifications 10-5
Validating Deposit Specifications 10-5
Getting Deposit Specifications 10-5
Modifying Deposit Specifications 10-6
Deleting Draft Deposit Specifications 10-6
Purchasing Deposits 10-6
Validating a Deposit Purchase 10-7
Getting Deposits 10-7
Updating Deposits 10-8
Validating Deposit Updates 10-8
Collecting Deposit Payments 10-8
Validating Deposit Payment Collections 10-9
Reversing Deposits 10-9
Validating a Deposit Reversal 10-10
Releasing Deposits 10-10

Validating a Deposit Release 10-11
Transferring Deposits 10-11
Validating a Deposit Transfer 10-11
Adding Interest to a Deposit 10-12
Customizing the Interest Calculation 10-12
Requesting Deposit Refunds 10-12
Validating a Deposit Refund Request 10-13
Getting Refund Requests 10-13
Updating Refund Requests 10-13
Updating Received Deposits 10-14
Triggering Deposits 10-14
Retrieving Transaction Details of a Deposit 10-14

11 Device Management Opcodes

Opcodes Described in This Chapter 11-1
About the Device Management Opcodes 11-2
Creating /device Objects 11-2
Changing Device Attributes and States 11-3
Changing the Attributes of /device Objects 11-4
Changing the State of a /device Object 11-4
Associating /service and /device Objects 11-6
Deleting /device Objects 11-7

12 General Ledger Opcode Workflows

Opcodes Described in This Chapter 12-1
How G/L IDs Affect Storage and Reporting of G/L Data 12-2
How BRM Records the Difference between Rounded and Unrounded Items 12-2
Customizing G/L Data Stored in /journal Objects 12-3
Customizing G/L Reports for Export 12-4
Customizing G/L Export Report Contents 12-4
Mapping BRM G/L Account Names to Third-Party Account Names 12-4
Mapping BRM Balance Element IDs to Third-Party Balance Element IDs 12-5
Customizing the G/L Report XML File Names 12-6
Assigning G/L IDs to Prerated Events 12-6
How BRM Exports G/L Reports 12-6
How BRM Stores General Ledger Reports 12-7

13 Group Opcode Workflows

Opcodes Described in This Chapter 13-1
Managing Sharing Groups 13-3
Creating a Sharing Group 13-4
Creating a Discount Sharing Group 13-5
Creating a Charge Sharing Group 13-6
Creating a Product Sharing Group 13-6
Creating a Balance Monitor Group 13-7
Validating the Members of a Balance Monitor Group 13-8
Modifying a Sharing Group 13-9
Adding Members to a Discount or Charge Sharing Group 13-9
Adding Discounts to a Discount Sharing Group 13-10
Adding Sponsored Charges to a Charge Sharing Group 13-11
Deleting a Sharing Group 13-11
Deleting a Discount Sharing Group 13-11
Deleting a Charge Sharing Group 13-12
Deleting a Member from a Discount Sharing Group 13-12
Deleting a Member from a Charge Sharing Group 13-12
Deleting a Shared Discount from a Discount Sharing Group 13-13
Deleting a Sponsored Charge from a Charge Sharing Group 13-13
Changing the Owner of a Sharing Group 13-13
Changing the Owner of a Discount Sharing Group 13-14
Changing the Owner of a Charge Sharing Group 13-14
Changing the Owner of a Balance Monitor 13-15
Adding a Monitor Group to a Member's /ordered_balgrp Object 13-15
Getting a List of Charges Available for Charge Sharing 13-16
Managing Ordered Balance Groups 13-17
Creating an Ordered Balance Group 13-18
Adding a Sharing Group to an Ordered Balance Group 13-18
Deleting a Sharing Group from an Ordered Balance Group 13-19
Customizing the Order in Which to Apply Sharing Groups 13-19
Deleting an Ordered Balance Group 13-19
Creating and Modifying Multiple Ordered Balance Groups Simultaneously 13-19
Adding and Removing Members Automatically to Sharing Groups 13-21
Adding Members to Newly Created Sharing Groups Automatically 13-21
Adding Members to Hierarchy-Type Balance Monitors 13-22
Adding Members to Payment Responsibility-Type Balance Monitors 13-23
Adding Members to Hierarchy-Type Sharing Groups Automatically 13-23
Adding Members to Paying Responsibility-Type Sharing Groups Automatically 13-24
Updating Subscription-Type Monitors Automatically 13-25
Removing Members from Balance Monitor Groups 13-25

Removing Members from Sharing Groups 13-26
Displaying Balance Monitor Information in Client Applications 13-26
Retrieving the Balances for a Monitor Group 13-27
Retrieving the Balance Monitors Owned by an Account or Service 13-27
Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications 13-27
Example of Credit Threshold Notification Event Generation 13-28
Managing Balance Monitors 13-29
Creating and Updating Balance Monitors 13-29
Deactivating Balance Monitors 13-30
Managing Balance Groups 13-30
Creating Balance Groups 13-30
Customizing the Default Balance Group of a Bill Unit 13-31
Moving a Balance Group from One Bill Unit to Another 13-32
Deleting a Balance Group 13-32
Modifying Sub-balances 13-32
Modifying the Sub-balance Validity Period 13-33
Converting an Account Hierarchy to Wholesale Billing 13-33
Rolling Up Non-Currency Balance from Members to the Owner of a Wholesale Sharing Group 13-34

14 Installment Opcode Workflows

Opcodes Described in This Chapter 14-1
Creating Installment Schedule Specifications 14-3
Validating Installment Schedule Specifications 14-4
Preparing Installment Schedule Specifications 14-4
Modifying Installment Schedule Specifications 14-4
Retrieving Installment Schedule Specifications 14-5
Getting Installment Schedule Specifications 14-5
Creating Installments 14-5
Validating Installments 14-5
Preparing Installments 14-6
Applying Installment Charge 14-6
Updating Installment Status 14-6
Retrieving Installments 14-6
Updating Installments 14-7
Canceling Installments 14-7
Installment Proposal 14-8

15 Invoice Opcode Workflows

Opcodes Described in This Chapter 15-1

Setting the Default Payment Due Date 15-3
Making Invoices 15-3
Displaying Invoices 15-5
Defining the Invoice Type 15-6
How Invoices Are Formatted 15-7
Customizing the Layout of Bill Items in Invoices 15-8
Customizing the Format for Printed Invoices 15-8
Customizing the Format for HTML Invoices 15-8
Customizing the Format for XML Invoices 15-9
Customizing the Invoice Format by Using an XSL Style Sheet 15-9
Customizing the Format for DOC1 Invoices 15-9
Displaying an Invoice On Demand 15-9
Including Custom Data in Invoices 15-10
Changing the Trial Billing Watermark 15-11
Including the Time Zone in Invoices 15-11
Adding Soft Descriptors to Invoices 15-11
Specifying Event Fields to Cache for Invoicing 15-12
Decoding Cached Event Data for Invoicing 15-15
How Invoices Are Generated 15-15
Customizing Invoice Search Operations 15-16
Creating Custom Search Templates 15-17
Example: Generating Invoices Based on Event Types 15-18
Configuring Error Checking for Customized Invoicing 15-19
Generating Trial Invoices 15-19
How Trial Invoicing Works for Hierarchical Bill Units in Oracle Analytics Publisher 15-19
About Associating Bill Units with an Oracle Analytics Publisher Invoice and Report 15-21
About the /associated_bus_profile Object 15-21
Creating /associated_bus_profile Objects 15-22

16 Job Opcode Workflows

Opcodes Described in This Chapter 16-1
Running BRM Applications 16-2
Running Business Operation Jobs 16-3
Managing Business Operations Job Templates 16-4
Managing Business Operations Jobs 16-5
Creating Custom Jobs 16-5
Modifying Custom Jobs 16-6
Deleting Custom Jobs 16-6
Validating Job Definitions 16-7
Post-Processing Job Definitions 16-7

17 Loan Opcode Workflows

Opcodes Described in This Chapter 17-1
Granting a Loan 17-2
Customizing Loan Grants 17-3
Granting a Channel Loan 17-3
Resetting a Loan Cycle and Other Loan Profile Fields 17-3
Checking a Customer's Eligibility for a Loan 17-3
Customizing Loan Eligibility Checks 17-5
Retrieving Loan Information 17-5
Creating and Modifying a Customer's Loan Profile 17-5
Customizing Loan Recovery 17-7

18 Notification Opcode Workflows

Opcodes Described in This Chapter 18-1
Creating Notification Specifications 18-2
Modifying Notification Specifications 18-2
Retrieving Notification Specifications 18-3
Managing In-Advance and Post-Expiration Notifications 18-3
Calculating the Notification Delivery Time 18-4
Verifying Customer Opt-In Preferences 18-4
Retrieving Last Notification Time and Offset 18-4
Customizing Notifications 18-5
Customizing Notification Retrieval 18-5
Preparing Notification Specifications 18-5
Validating Notification Specifications 18-5

19 Number Manager and SIM Card Manager Opcode Workflows

Opcodes Described in This Chapter 19-1
Number Manager Opcodes 19-2
Creating Blocks of Numbers 19-3
Modifying Blocks of Numbers 19-3
Splitting Blocks of Numbers 19-3
Managing Number Quarantine 19-3
Managing Number Portability 19-4
Customizing Number Manager 19-5
Customizing Number Normalization 19-5
Customizing How Numbers Are Associated with Services 19-5
Customizing Telephone Number Attributes 19-6
Customizing How a Number Can Be Changed 19-6

Deleting a Number 19-7
SIM Card Manager Opcodes 19-7
Creating and Updating SIM Card Orders 19-7
Creating SIM Cards 19-7
Provisioning SIM Cards 19-8
Customizing SIM Card Manager 19-8
Customizing SIM Card Service Association 19-8
Customizing SIM Card Validation 19-9
Customizing SIM Card Number Changes 19-9

20 Role Opcode Workflows

Opcodes Described in This Chapter 20-1
Creating a Role 20-1
Modifying a Role 20-3
Deleting a Role 20-4
Associating a pcm_client or admin_client Service with a Role 20-4

21 Payment Opcode Workflows

Opcodes Described in This Chapter 21-1
Collecting Payments 21-7
Finding Payments 21-8
BRM-Initiated Payment Processing 21-8
Initiating a Charge 21-11
Processing Credit Card Charges 21-12
Processing the Results of Credit Card Transactions 21-13
Processing Paymentech Address Validation Return Codes 21-15
Processing AVS Validations for International Credit Cards 21-16
Processing Direct Debit Charges 21-16
About Paymentech Direct Debit Implementation 21-17
Creating a Custom Direct Debit Implementation 21-17
Processing a Batch of Direct Debit Charges 21-18
Checking the Results of BRM-Initiated Batch Payment Operations 21-18
Validating Credit Card and Direct Debit Transactions 21-19
Processing Credit Card Information during Account Creation 21-19
Storing Card Credentials for Future Transactions 21-20
Purging Card Credentials 21-21
About Credit Card Payment Confirmation Numbers 21-21
Externally Initiated Payment Processing 21-21
Receiving Payments 21-22
Validating Payments 21-23

About the Default Payment Validation Process 21-23
About Payment Validation Flags 21-26
Configuring Unconfirmed Payment Processing 21-26
Handling Overpayments and Underpayments 21-27
Calculating Payment Collection Dates 21-28
How BRM Selects the Items to Which Payments Are Applied 21-29
Allocating Account Payments to Multiple Bill Units 21-31
Allocating Externally Initiated Payments by Due Amount 21-33
Reversing Payments 21-33
Refunding Payments 21-35
Writing Off Payments 21-36
Finding Payment Info 21-36
Adding a Custom Payment Method 21-36
Creating a /config/payment Object 21-37
Viewing Payment Information for a Custom Payment Method 21-37
Configuring Payment Center for Custom Payment Methods 21-37
How Opcodes Read the Payment Method 21-38
Processing Payment Fees 21-40
Applying Payment Fees 21-40
Customizing Payment Fees 21-41
Using Custom Reason Codes to Customize Payment Fees 21-42
Customization example: Charging a fee based on the customer segment 21-43
Storing Additional Information with Payment Fees 21-43
Processing Payment Incentives 21-43
Triggering Payment Incentives 21-44
Granting Payment Incentives 21-45
Reversing Payment Incentives 21-45
Customizing Payment Incentives 21-46
Customizing How to Trigger Payment Incentives 21-46
Customizing How to Grant Payment Incentives 21-46
Manually Reversing a Payment Incentive 21-47
Managing Top-Ups 21-48
Setting Up Top-Up Information in an Account 21-48
Preparing an Account's Top-Up Information 21-49
Validating an Account's Top-Up Information 21-50
Finding Top-Up Events 21-51
Creating an Account's Top-Up Information 21-51
Modifying an Account's Top-Up Information 21-52
Deleting Top-Ups 21-52
Finding Sponsored Top-Up Groups 21-52
Setting a Customer's Automatic Top-Up Threshold 21-53
Processing Top-Ups 21-53

Triggering Top-ups 21-54
How BRM Handles Manual Standard Top-Ups 21-54
How BRM Handles Automatic Standard Top-Ups 21-55
How BRM Handles Recurring Standard Top-Ups 21-56
How BRM Handles Manual Sponsored Top-Ups 21-57
How BRM Handles Automatic Sponsored Top-Ups 21-58
Implementing Top-Ups in Custom Client Applications 21-59
Implementing Manual Standard Top-Ups 21-59
Implementing Automatic Standard Top-Ups 21-60
Implementing Recurring Standard Top-Ups 21-61
Implementing Manual Sponsored Top-Ups 21-62
Implementing Automatic Sponsored Top-Ups 21-64
Viewing Sponsored Top-Up History 21-64
About Transferring Sponsored Top-Ups from Debit Balances 21-66
Setting an Account's Sponsored Top-Up Member Status and PIN 21-66
Activating Sponsored Top-Up Group Members 21-66
Inactivating Sponsored Top-Up Group Members 21-67
Setting Sponsored Top-Up Member PINs 21-67
Offering Discount Incentives with Top-Ups 21-67
Suspending and Recycling Payments 21-68
How BRM Tracks Suspended Payments 21-68
Customizing Payment Suspense 21-70
Customizing Payment Suspense Validation 21-71
Customizing Suspended Payment Distribution 21-73
Customizing Payment Failure Reason Codes 21-73
Handling Custom Payment Methods 21-73
About the Suspense Account 21-74
Suspending Payments during Payment Processing 21-74
How Payments Are Recycled to and from Suspense 21-75
Understanding Payment Recycling 21-77
About Original Payments 21-77
About Payment Transfer Direction and Verification 21-78
About Recycling Payments from Suspense 21-78
About Recycling Payments to Suspense 21-78
How Payment Reversals Work with Suspense and Recycling 21-79
About Directly Reversing Payments from BRM 21-79
Retrieving Recycled Payments 21-80
How Suspended Payments Are Reversed 21-81
How Payments Are Reversed During Recycling 21-81
How Payments Are Removed As Unallocatable 21-81
How BRM Handles Mandate Information for SEPA Processing 21-82
How BRM Registers a Mandate 21-82

How BRM Updates a Mandate 21-82
Updating Creditor Information 21-83
How BRM Cancels a Mandate 21-83

22 Subscription Opcode Workflows

Opcodes Described in This Chapter 22-1
Pricing Component Object Names 22-6
Managing Charge Offers 22-6
Purchasing Charge Offers 22-7
Applying Deferred Charge Offer Purchase Fees 22-9
About Delayed Cycle Start and End Times 22-10
Handling Purchase, Cycle, and Usage Start and End Times 22-10
Purchasing the Same Charge Offer Multiple Times 22-11
Backdating Charge Offer, Discount Offer, Bundle, or Package Purchases 22-12
How Sub-balance Buckets Are Created for Backdated Charge Offer Purchases 22-12
How Cycle Fees Are Calculated for Backdated Purchase or Activation 22-13
Applying Purchase and Cycle Fees to the Balance 22-13
How Charge Offers Are Modified 22-14
How BRM Changes Charge Offer Status 22-14
How Charge Offers Are Canceled 22-16
Customizing Charge Offer Cancellation 22-18
Customizing How to Apply Folds 22-18
Managing Discount Offers 22-19
How Discount Offers Are Purchased 22-19
How Discount Offer Purchase, Cycle, and Usage Validity Is Modified 22-22
How BRM Changes Discount Offer Status 22-23
How Discount Offers Are Canceled 22-23
Validating Discount Offer Dependencies 22-25
About Optional and Required Service Types 22-26
About Closing a Required Service 22-26
Customizing Snowball Discounts 22-26
Customizing Discount Offer Cancellation 22-26
Managing Bundles 22-27
How Bundles Are Purchased 22-27
Validating Changes to Bundles 22-30
How Bundles Are Modified 22-30
How Bundles Are Transitioned 22-31
Customizing Bundle Transitions 22-31
How Bundle Dependencies Are Validated 22-32
Validating Bundle Transitions 22-33
How Bundles Are Canceled 22-33

Overriding Deal Settings at Customer Level 22-34
Overriding Add-On Product Validity Dates in a Deal 22-34
Overriding Bundle Proration Settings at Customer Level 22-35
Managing Packages 22-35
How BRM Transitions Accounts from Source Packages to Target Packages 22-36
Transition Type API Considerations 22-38
Customizing Package Transitions without Configuring /transition Objects 22-38
About Providing Transition Rules Using Policy Opcodes 22-39
Customizing Account Bundles during Package Transitions 22-40
Managing Contracts 22-40
Creating Contracts 22-41
Modifying Contracts 22-41
Canceling Contracts 22-41
Renewing Contracts 22-41
Extending Subscriber Contracts 22-42
Managing Purchase, Cycle, and Usage Charges in Purchased Offers 22-42
Changing the Purchase, Usage, and Cycle Start and End Times 22-44
Overriding Charges and Discounts for a Period of Time 22-45
Specifying Purchase, Cycle, and Usage Start and End Times 22-46
Calculating the Cycle Forward Fee 22-47
Calculating the Cycle Arrears Fee 22-47
Cycle Arrears Charge Offer Limitations 22-48
About Backdated Purchase, Usage, or Cycle End Dates 22-48
Storing Relative Start and End Times for an Account's Charge Offers and Discount Offers 22-50
How Cycle Fees Are Calculated for Backdated Cancellation or Inactivation 22-51
How Earned and Unearned Revenue Is Set for Backdated Period 22-51
Applying Recurring Charges 22-52
Applying Cycle Forward Fees 22-52
Applying Cycle Arrears Fees 22-53
Applying Folds 22-53
Customizing How Folds Are Applied 22-54
Customizing the Order to Apply Folds 22-54
Specifying Which Balances to Fold 22-54
Customizing Which Balances to Fold When Charge Offers Are Canceled 22-54
Getting Data about Bundles, Charge Offers, Discount Offers, and Services 22-55
Getting Purchased Offers 22-55
Getting Packages, Bundles, and Charge Offers for Purchase 22-57
Getting a List of Bundles, Charge Offers, Discount Offers, and Services 22-58
Getting a List of Packages and Bundles That an Account Owns 22-58
Reading Data for All Valid Purchased Charge Offers and Discount Offers 22-59
Finding Events Associated with Bundles, Charge Offers, Discount Offers, and Services 22-61

Getting a Life-Cycle State 22-61
Getting Information about Price Tags 22-62
Applying Promotions for Special Dates, Events, or Actions 22-65
Customizing Special Date, Event, or Action Promotions 22-66
Managing Promotions with Siebel CRM 22-66
Managing Provisioning 22-68
Customizing Provisioning When a Charge Offer Is Purchased 22-69
Getting a List of Provisioning Tags 22-69
Customizing Provisioning When Canceling a Charge Offer 22-70
Managing GSM Service Provisioning 22-70
Validating Product Specification Attributes for Pricing Components 22-71

23 Tax Calculation Opcode Workflows

Opcodes Described in This Chapter 23-1
Calculating Taxes during Billing 23-2
How PCM_OP_RATE_TAX_CALC Calculates Taxes 23-3
How PCM_OP_RATE_TAX_CALC Validates Addresses and VAT Certificates 23-4
Policy Opcodes Called by PCM_OP_RATE_TAX_CALC 23-5
Modifying Tax Data Before Calculating Taxes 23-5
Customizing Vertex Communications Tax Q Series to Override ZIP Codes 23-5
Customizing Vertex Communications Tax Q Series to Provide Custom Input Tax Data 23-6
Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX 23-6
Modifying Tax Data After Calculating Taxes 23-10
Overriding Customer Tax Locale for Purchases 23-10
Using Custom Tax Rates 23-12
Using Geocodes to Calculate Taxes 23-13
Adding Tax Information to Accounts 23-14
Validating Tax Information 23-14
Retrieving Tax Calculation Data 23-15
Retrieving a List of Tax Codes 23-15
Retrieving a List of Tax Suppliers 23-15
Retrieving Tax Supplier Data 23-16
Retrieving Tax Location Data 23-16
Retrieving Additional Tax Data from Vertex Communications Tax Q Series 23-17
Default Tax Data Returned by Vertex Communications Tax Q Series 23-18
Requesting Additional Data from Vertex Communications Tax Q Series 23-19
Retrieving Additional Tax Data 23-20
Storing the Requested Vertex Tax Data in the BRM Database 23-22
Creating a Custom Storable Class for Vertex Tax Data 23-22
Storing Vertex Tax Data in Your Custom Storable Class 23-23

24 Using the IP Address Manager APIs

Opcodes Described in This Chapter 24-1
Managing Your IP Address Device Life Cycle 24-2
- Creating a Single IP Address Device 24-2
- Creating a Range of IP Address Devices 24-3
- Creating a Range of IP Addresses with a Subnet Mask 24-4
- Customizing IP Address Creation 24-4
- Associating an IP Address with Accounts or Services 24-4
- Disassociating an IP Address Device from Accounts or Services 24-4
- Changing IP Device States from Unallocated to Returned 24-4
- Modifying an IP Address Device 24-5
- Sorting IP Devices by Using Canonical IP Address 24-5
- Deleting an IP Address Device 24-6

Managing your APN Device Life Cycle 24-6
- Creating an APN Device 24-6
- Associating APN with an Account or Service 24-6
- Modifying an APN Device 24-7
- Changing the APN Device State 24-7
- Deleting an APN Device 24-8

Extending the IP Address Manager Storable Classes 24-8
Adding Business Logic to the IP Address and APN Policy FMs 24-8

Preface

This guide provides workflow and reference information about Oracle Communications Billing and Revenue Management (BRM) opcodes.

Audience

This guide is intended for developers.

Documentation Accessibility

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

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

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

Part I
Opcode Reference

This part provides Oracle Communications Billing and Revenue Management (BRM) opcode reference information.

Part I contains the following chapters:

- Opcode Descriptions
- Context Management Opcodes

1

Opcode Descriptions

Learn about the Oracle Communications Billing and Revenue Management (BRM) opcodes.

About Using Opcodes

For information about using opcodes, see the following discussions in *BRM Developer's Guide*:

- About customizing BRM
- Writing a custom Facilities Module (FM)
- Understanding the Portal Communications Module (PCM) application programming interface (API) and the Portal Information Network (PIN) library
- Understanding API error handling and logging
- Header files

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

You can find information about the input and output fields required for each opcode in the opcode flist specifications. See *BRM Opcode Flist Reference* for more information.

Each opcode flist specification also specifies the following information:

- The opcode's visibility. See "About the Opcode Visibility Value".
- The opcode's transaction handling. See "About the Opcode Transaction Value".

About the Opcode Visibility Value

An opcode's visibility field can have one of these values:

- **Public**: This is a recommended opcode. Oracle will strive to keep the interface intact.
- **Private**: Call this opcode only if absolutely necessary. This opcode may change from release to release.
- **Deprecated**: This opcode is deprecated. Either the functionality is being discontinued or other opcodes may provide equivalent or better functionality.
- **Obsolete**: This opcode is obsolete and should no longer be used. It is no longer supported.

About the Opcode Transaction Value

An opcode's transaction field can have one of these values:

- **Required**: The transaction for this opcode can be wrapped in a transaction opened by another opcode.

  If a read-write transaction is already open when this opcode is run, all data modifications take place within the open transaction. The modifications are committed or canceled along with all other changes when the transaction is committed or canceled.

  If no transaction is open when the opcode is called, a read-write transaction is opened. All actions are performed within this transaction, ensuring that the entire operation is performed atomically. If an error occurs during the execution of the opcode, all changes are canceled when the transaction is canceled. If no error occurs, the transaction is committed at the end of the operation.

  This opcode requires a read-write transaction. It is therefore an error to have a read-only transaction open when this opcode is called.

- **Requires New**: This opcode manages transactions internally to ensure absolute integrity of the database. A transaction for this opcode cannot be wrapped in another transaction.

  If no transaction is open when the opcode is called, a read-write transaction is automatically opened and all actions are performed within this transaction.

  If a transaction is already open when the opcode is called, an error occurs.

- **Supports**: This opcode does not modify object data. If it is called while a transaction is not already open, the operation is run without transactional control.

  If a read-write or read-only transaction is already open when this opcode is called, the opcode is run as part of the transaction and reads the in-process state of the data.

  If the opcode is called when a separate, unrelated transaction is taking place, it reads the last saved state of the database.

Account Dump FM Policy Opcodes

Use the account dump policy opcodes to customize Account Dump Utility (ADU) validation and output file format. The Account Dump utility (ADU) is a diagnostics tool that enables you to validate account information before or after certain business processes (for example, after completion of a migration or upgrade or before billing or payment allocation).

See *BRM Customer Management* for information about ADU.

Include the **ops/cust.h** header file in all applications that call these opcodes.

Table 1-1 Account Dump FM Policy Opcodes

Opcode	Description
PCM_OP_ADU_POL_DUMP	Recommended. Enables you to convert the ADU output format to a format other than XML. See "Customizing the Account Dump Utility (ADU)".

Table 1-1 (Cont.) Account Dump FM Policy Opcodes

Opcode	Description
PCM_OP_ADU_POL_VALIDATE	Recommended. Enables you to define custom validations for validating account data. See "Customizing the Account Dump Utility (ADU)".

Account Dump FM Standard Opcode

Use the account dump policy opcodes to customize Account Dump Utility (ADU) validation and output file format. The Account Dump utility (ADU) is a diagnostics tool that enables you to validate account information before or after certain business processes (for example, after completion of a migration or upgrade or before billing or payment allocation).

See *BRM Customer Management* for information about ADU.

Include the **ops/cust.h** header file in all applications that call these opcodes.

Table 1-2 Account Dump FM Standard Opcode

Opcode	Description
PCM_OP_ADU_VALIDATE	Recommended. Dumps the contents of objects for an account into an output file and validates the contents of the output file.

Accounts Receivable FM Policy Opcodes

Use the accounts receivable policy opcodes to manage accounts receivable (A/R) functions such as adjustments, disputes, and write-offs.

Include the **ops/ar.h** header file in all applications that call these opcodes.

Table 1-3 Accounts Receivable FM Policy Opcodes

Opcode	Description
PCM_OP_AR_POL_PRE_EVENT_ADJUSTMENT	Recommended. Enables you to customize the input flist by adding or deleting events. Customizing the input flist allows you to include amounts from events into the total amount available for adjustments.
PCM_OP_AR_POL_REVERSE_WRITEOFF	Recommended. Enables you to customize how to retrieve write-off reversal items from the **/profile/writeoff** object. See "Reversing Write-Offs".

Accounts Receivable FM Standard Opcodes

The accounts receivable standard opcodes take PIN_FLD_END_T as an optional field in the input flist to determine when an A/R operation needs to be done. If this field is not passed in the input flist, the A/R operation is carried out with the current time. If PIN_FLD_END_T is

specified, the date must be earlier than or equal to the current date. BRM does not support future dating an A/R operation.

Include the **ops/ar.h** header file in all applications that call these opcodes.

Table 1-4 Accounts Receivable FM Standard Opcodes

Opcode	Description
PCM_OP_AR_ACCOUNT_ADJUSTMENT	Recommended. Debits or credits an account balance for currency adjustments. See "Adjusting Accounts, Subscription Services, and Member Services".
PCM_OP_AR_ACCOUNT_WRITEOFF	Recommended. Performs write-off of one or more A/R bill units (**/billinfo** objects) for an account. See "About Initiating Write-Offs" and "About Account Write-Offs".
PCM_OP_AR_BILL_ADJUSTMENT	Recommended. Credits the currency balance of an account's AR bill. See "Adjusting Bills".
PCM_OP_AR_BILL_CREDIT_TRANSFER	Recommended. Transfers the amount from a bill that has a negative balance to one or more bills that have a positive balance. See "Transferring from a Negative to a Positive Balance".
PCM_OP_AR_BILL_DISPUTE	Recommended. Opens a dispute against the account's A/R bill. See "Disputing Bills".
PCM_OP_AR_BILL_SETTLEMENT	Recommended. Settles an A/R bill that is in dispute. See "Settling Disputed Bills".
PCM_OP_AR_BILL_WRITEOFF	Recommended. Performs write-off adjustments against a specified bill. See "About Bill Write-Offs".
PCM_OP_AR_BILLINFO_WRITEOFF	Recommended. Performs write-off adjustments for a specific bill unit. See "About Bill Unit Write-Offs".
PCM_OP_AR_EVENT_ADJUSTMENT	Recommended. Adjusts the balance impact of an event on an account's balance. See "Adjusting Events".
PCM_OP_AR_EVENT_DISPUTE	Recommended. Opens a dispute against one or more events associated with an account. See "Disputing Events".
PCM_OP_AR_EVENT_SETTLEMENT	Recommended. Settles dispute events associated with an account. See "Settling Disputed Events".
PCM_OP_AR_GET_ACCT_ACTION_ITEMS	Recommended. Retrieves the list of A/R items applied to all bill units in an account or to a single bill unit. See "Retrieving A/R Items That Apply to a Bill Unit".

Table 1-4 (Cont.) Accounts Receivable FM Standard Opcodes

Opcode	Description
PCM_OP_AR_GET_ACCT_BAL_SUMMARY	Recommended. Retrieves the consolidated unapplied, open bill due, pending bill due, and total dispute balances for all the bill units in an account or for a specified bill unit in an account. See "Retrieving a Balance Summary".
PCM_OP_AR_GET_ACCT_BILLS	Recommended. Retrieves the list of bills for all bill units in an account or for a single bill unit. See "Retrieving a List of Bills for a Bill Unit".
PCM_OP_AR_GET_ACTION_ITEMS	Recommended. Retrieves the list of A/R items applied to a bill unit. See "Returning a Limited Set of Dispute Data" and "Retrieving Dispute Details for a Bill Unit".
PCM_OP_AR_GET_BAL_SUMMARY	Recommended. Retrieves the applied, unapplied, open bill due, pending bill due, and total dispute balances for a specified bill unit. See "Retrieving a Balance Summary".
PCM_OP_AR_GET_BILL_ITEMS	Recommended. Retrieves the list of bill items for a bill unit. See "Retrieving a List of Bills for a Bill Unit".
PCM_OP_AR_GET_BILLS	Recommended. Retrieves a list of bills for a bill unit based on the start time and end time search criteria. See "Retrieving a List of Bills for a Bill Unit".
PCM_OP_AR_GET_DISPUTE_DETAILS	Recommended. Retrieves all event and item disputes and the aggregated disputed amount for the events associated with a dispute item (**/item/dispute** object). See "Retrieving Dispute Details for a Bill Unit".
PCM_OP_AR_GET_DISPUTES	Recommended. Retrieves details of all disputed bill items for a bill unit. See "Retrieving Dispute Details for a Bill Unit".
PCM_OP_AR_GET_ITEM_DETAIL	Recommended. Retrieves details for the specified A/R item or bill item. See "Retrieving Details about a Specific A/R Item or Bill Item".
PCM_OP_AR_GET_ITEMS	Recommended. Retrieves details for the specified A/R item or bill item. See "Retrieving Details about a Specific A/R Item or Bill Item".
PCM_OP_AR_ITEM_ADJUSTMENT	Recommended. Makes adjustments against an item. See "Adjusting Items".
PCM_OP_AR_ITEM_DISPUTE	Recommended. Files a dispute against an item. See "Disputing Items".

Table 1-4 (Cont.) Accounts Receivable FM Standard Opcodes

Opcode	Description
PCM_OP_AR_ITEM_SETTLEMENT	Recommended. Settles an item that is in dispute. See "Settling Disputed Items".
PCM_OP_AR_ITEM_WRITEOFF	Recommended. Performs a write-off adjustment of an item. See "Performing Write Offs".
PCM_OP_AR_RESOURCE_AGGREGATION	Limited. Calculates the aggregated amount of event balance impacts for each balance affected by the event. See "Retrieving Details about a Specific A/R Item or Bill Item".
PCM_OP_AR_REVERSE_REFUND	Limited. Reverses failed refunds on accounts. See "Reversing Refunds".
PCM_OP_AR_REVERSE_WRITEOFF	Limited. Reverses write-offs on all written-off bills and bill items associated with a written-off account or bill unit when a payment for the account or bill unit is received. See "Reversing Write-Offs".

Account Synchronization FM Opcodes

The account synchronization opcodes synchronize BRM and Elastic Charging Engine (ECE) data. BRM events that trigger event notification for account synchronization make up the business events that the Oracle Data Manager (DM) sends to ECE via an Oracle DM database queue.

See *BRM System Administrator's Guide* for information about account synchronization.

Include the **ops/ifw_sync.h** header file in all applications that call these opcodes.

Table 1-5 Account Synchronization FM Opcodes

Opcode	Description
PCM_OP_IFW_SYNC_PUBLISH_EVENT	Limited Passes events associated with this opcode in your system's event notification list to PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT for processing. PCM_OP_IFW_SYNC_PUBLISH_EVENT does not modify events.
PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT	Recommended. Enables you to modify account synchronization events before they are sent to ECE. See "Modifying Events before Sending Them to ECE".

Activity FM Policy Opcodes

Uses the activity policy opcodes to manage event creation, event recording, and event notification.

Include the **ops/act.h** header file in all applications that call these opcodes.

Table 1-6 Activity FM Policy Opcodes

Opcode	Description
PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE	Recommended. Enables you to specify in which billing cycle to apply an event when an event occurs between the end of a billing cycle and when billing applications are run. See "Customizing How to Bill Events That Occur between Billing Cycles".
PCM_OP_ACT_POL_EVENT_LIMIT	Recommended. Enables you to inactivate an account or account hierarchy, and send a notification that a limit has been reached. See "Inactivating Accounts that Exceed a Specified Limit".
PCM_OP_ACT_POL_EVENT_NOTIFY	Recommended. Enables you to implement custom event notification. See "Using Event Notification" in *BRM Developer's Guide*.
PCM_OP_ACT_POL_LOCK_SERVICE	Recommended. Enables you to customize how to lock an account for **/service/pcm_client** and **/service/admin_client** after a specified number of invalid login attempts. See "Managing Login Names and Passwords" in *BRM System Administrator's Guide*.
PCM_OP_ACT_POL_LOG_USER_ACTIVITY	Recommended. Enables you to add details for the events that must be logged. See "Logging Customer Service Representative Activity Events" in *BRM System Administrator's Guide*.
PCM_OP_ACT_POL_PROCESS_EVENTS	Recommended. Processes events and updates the database with news feed entries. See "Customizing News Feed Values and Events".
PCM_OP_ACT_POL_REQUEST_CREATE	Recommended. Enables you to customize operation failure records before they are stored in the database. See "Recording Failed Operations".
PCM_OP_ACT_POL_SPEC_EVENT_CACHE	Recommended. Enables you to define which balance impact fields are cached for invoicing. See "Specifying Event Fields to Cache for Invoicing".
PCM_OP_ACT_POL_SPEC_GLID	Recommended. Enables you to customize how BRM assigns a G/L ID to an event. See "Assigning G/L IDs for an Adjustment".

Table 1-6 (Cont.) Activity FM Policy Opcodes

Opcode	Description
PCM_OP_ACT_POL_SPEC_RATES	Recommended. Enables you to implement custom rating. See "Implementing Custom Rating".
PCM_OP_ACT_POL_SPEC_VERIFY	Recommended. Enables you to customize authentication checks. See "Customizing Authentication Checks".
PCM_OP_ACT_POL_VALIDATE_SCHEDULE	Recommended. Enables you to customize policy checks before creating **/schedule** objects. See "Performing Policy Checks before Scheduling Deferred Actions".

Activity FM Standard Opcodes

The activity standard opcodes manage sessions.

Include the **ops/act.h** header file in all applications that call these opcodes.

Table 1-7 Activity FM Standard Opcodes

Opcode	Description
PCM_OP_ACT_END_SESSION	Recommended. Logs the end of a session event. See "Recording the End of a Session " and "Recording Session Events".
PCM_OP_ACT_FIND	Recommended. Finds a customer's account. See "Finding a Customer's Account Information".
PCM_OP_ACT_FIND_VERIFY	Recommended. Authenticates a user action. See "Starting Sessions".
PCM_OP_ACT_LOAD_SESSION	Recommended. Creates and records a session event as a single operation. See "Loading Sessions".
PCM_OP_ACT_LOGIN	Recommended. Authorizes users to start login sessions. See "Starting Sessions".
PCM_OP_ACT_LOGOUT	Recommended. Records the end of a login session. See "Recording the End of a Session ".
PCM_OP_ACT_REQUEST_CREATE	Recommended. Creates a **/request** object, which records information about a failed operation. See "Recording Failed Operations".

Table 1-7 (Cont.) Activity FM Standard Opcodes

Opcode	Description
PCM_OP_ACT_REQUEST_RETRIEVE	Recommended. Retrieves **/request** objects. See "Retrieving Operation Failure Records".
PCM_OP_ACT_SCHEDULE_CREATE	Recommended. Creates a **/schedule** object, which defers a single action to a predetermined date and time. See "Scheduling Deferred Actions".
PCM_OP_ACT_SCHEDULE_DELETE	Recommended. Deletes a **/schedule** object. See "Managing Deferred Actions".
PCM_OP_ACT_SCHEDULE_EXECUTE	Recommended. Runs any deferred actions in specified **/schedule** objects. See "Executing Deferred Actions ".
PCM_OP_ACT_SCHEDULE_MODIFY	Recommended. Modifies the text description of an existing **/schedule** object. See "Modifying Deferred Actions".
PCM_OP_ACT_START_SESSION	Recommended. Creates session events and records their start times. See "Recording the Start of a Session".
PCM_OP_ACT_UPDATE_SESSION	Recommended. Updates session event information. See "Updating a Session Event".
PCM_OP_ACT_USAGE	Last resort Manages session events. See "Managing Sessions".

Balance FM Policy Opcodes

Use the balance policy opcodes to trigger service life cycle state changes based on balance adjustments or to customize algorithms to select the default balance group of a bill unit.

Include the **ops/bal.h** header file in all applications that call these opcodes.

Table 1-8 Balance FM Policy Opcodes

Opcode	Description
PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS	Recommended. Enables you to customize threshold processing for policy-driven charging. When you start your BRM system, BRM caches the set of unique balance element IDs that you configured in your offer profiles. This opcode processes the threshold calculation logic only if the balance element ID is one of the entries in its cache. This opcode is called by PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS.

Table 1-8 (Cont.) Balance FM Policy Opcodes

Opcode	Description
PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	Recommended. Enables you to customize how to get life-cycle states. See "Creating Custom Service Life Cycles" in *BRM Managing Customers*.
PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC	Recommended. Enables you to customize how to select the default balance group of a bill unit and the default service of the default balance group. See "Customizing the Default Balance Group of a Bill Unit".
PCM_OP_BAL_POL_SET_SUB_BALANCES	Recommended. Enables you to customize an account's or service's sub-balance before it is transferred. See "Customizing Balance Transfers".

Balance FM Standard Opcodes

The balance standard opcodes adjust account balances.

Include the **ops/bal.h** header file in all applications that call these opcodes.

Table 1-9 Balance FM Standard Opcodes

Opcode	Description
PCM_OP_BAL_CHANGE_VALIDITY	Recommended. Changes a sub-balance's validity period. See "Modifying the Sub-balance Validity Period".
PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC	Recommended. Returns the balance groups and services for all the account's bill units or for a single bill unit. See "Finding a Balance Group and Service for Bill Units".
PCM_OP_BAL_GET_ACCT_BILLINFO	Recommended. Returns the main contact information for an account and a list of the account's bill units with the default bill unit marked. See "Finding a Bill Unit".
PCM_OP_BAL_GET_BALANCES	Recommended. Returns the POID of a **/balance_group** object and, optionally, the balances it contains. See "Finding a Balance Group and Its Balances".
PCM_OP_BAL_GET_BAL_GRP_AND_SVC	Recommended. Gets the balance groups and services for a bill unit See "Finding a Balance Group and Service for Bill Units".
PCM_OP_BAL_GET_ECE_BALANCES	Recommended. Gets the real-time balances for a service from ECE. This opcode can be called by a custom application.

Table 1-9 (Cont.) Balance FM Standard Opcodes

Opcode	Description
PCM_OP_BAL_TRANSFER_BALANCE	Recommended. Transfers currency and non-currency balances from one account or service to another. See "Transferring Balances between Accounts or Services".

Base Opcodes

The base opcodes can be used by any opcode. Unlike all other opcodes, which belong to the Connection Manager, the base opcodes are run by data managers.

Each DM included with BRM uses a different implementation of the base opcodes. For example, the base opcode PCM_OP_SEARCH is implemented differently for the Oracle DM and the LDAP DM. For information, see:

- LDAP Base Opcodes
- Email Data Manager Opcodes

Include the **ops/base.h** header file in all applications that call these opcodes.

Table 1-10 Base Opcodes

Base Opcodes	Description
PCM_OP_BULK_CREATE_OBJ	Recommended. Creates a large number of objects of the same type. See the discussion on creating a large number of objects in *BRM Developer's Guide*.
PCM_OP_BULK_DELETE_OBJ	Recommended. Deletes a large number of objects of the same type. See the discussion on deleting a large number of objects in *BRM Developer's Guide*.
PCM_OP_BULK_WRITE_FLDS	Recommended. Updates the value of the same fields in a large number of objects that meet the conditions you specify in the query in the input flist. See the discussion on editing large numbers of objects in *BRM Developer's Guide* and "Editing Suspended Records in Bulk" for more information.
PCM_OP_CREATE_OBJ	Recommended. Creates an object. See the discussion on creating objects in *BRM Developer's Guide*.
PCM_OP_DELETE_FLDS	Recommended. Deletes arrays or array elements from an object. See the discussion on deleting fields in objects in *BRM Developer's Guide*.
PCM_OP_DELETE_OBJ	Recommended. Deletes a specified object and all its fields. See the discussion on deleting objects in *BRM Developer's Guide*.

Table 1-10 (Cont.) Base Opcodes

Base Opcodes	Description
PCM_OP_GET_DD	Recommended. Retrieves the Data Dictionary. Oracle recommends using the **pin_deploy** utility to invoke this opcode. See the discussion on the **pin_deploy** utility in *BRM Developer's Guide*.
PCM_OP_GET_PIN_VIRTUAL_TIME	Recommended. Retrieves the virtual time that is set in the BRM Connection Manager (CM). Use this opcode **only** in a test environment when you want to retrieve the virtual time for a custom application that is connected to the CM. Oracle recommends using the **pin_virtual_time** utility to invoke this opcode. See the discussion on the **pin_deploy** utility in *BRM Developer's Guide*.
PCM_OP_GLOBAL_SEARCH	Recommended. To perform a global search, use PCM_OP_GLOBAL_SEARCH. See the discussion on performing a global search in *BRM Developer's Guide*.
PCM_OP_GLOBAL_STEP_END	Recommended. Ends global step-searching initiated by PCM_OP_GLOBAL_STEP_SEARCH. See the discussion on performing a global step search in *BRM Developer's Guide*.
PCM_OP_GLOBAL_STEP_NEXT	Recommended. Receives the next set of search results from a step search. See the discussion on getting the next set of search results from a global step search in *BRM Developer's Guide*.
PCM_OP_GLOBAL_STEP_SEARCH	Recommended. Step-searches for objects across multiple BRM database schemas. See the discussion on performing a global step search in *BRM Developer's Guide*.
PCM_OP_INC_FLDS	Recommended. Increments or decrements one or more fields in an object. See the discussion on incrementing fields in objects in *BRM Developer's Guide*.
PCM_OP_READ_FLDS	Recommended. Reads one or more fields in an object. See the discussion on reading fields in an object *BRM Developer's Guide*.
PCM_OP_READ_OBJ	Recommended. Reads an entire object. See the discussion on reading an entire object in *BRM Developer's Guide*.
PCM_OP_SEARCH	Recommended. Searches for objects in a single BRM database schema. See the discussion on performing single-schema searches in *BRM Developer's Guide*.

Table 1-10 (Cont.) Base Opcodes

Base Opcodes	Description
PCM_OP_SET_DD	Recommended. Modifies the Data Dictionary. Oracle recommends using the **pin_virtual_time** utility to invoke this opcode. See the discussion on the **pin_deploy** utility in *BRM Developer's Guide*.
PCM_OP_STEP_END	Recommended. Ends a step search initiated by PCM_OP_STEP_SEARCH. See the discussion on ending a step search in *BRM Developer's Guide*.
PCM_OP_STEP_NEXT	Recommended. Retrieves the next set of search results from a step search. See the discussion on getting the next set of search results from a step search in *BRM Developer's Guide*.
PCM_OP_STEP_SEARCH	Recommended. Step-searches for objects in a BRM database schema. See the discussion on performing single-schema step searches in *BRM Developer's Guide*.
PCM_OP_TEST_LOOPBACK	Recommended. Tests directory server connections. See the discussion on the **noop testnap** command in the chapter about testing applications in *BRM Developer's Guide*.
PCM_OP_TRANS_ABORT	Recommended. Ends an open PCM transaction. See the discussion on cancelling transactions in *BRM Developer's Guide*.
PCM_OP_TRANS_COMMIT	Recommended. Commits an open transaction on a PCM context. See the discussion on committing transactions in *BRM Developer's Guide*.
PCM_OP_TRANS_OPEN	Recommended. Opens a transaction. See the discussion on using transactions in *BRM Developer's Guide*.
PCM_OP_TRANS_POL_ABORT	Recommended. Enables you to customize how to cancel a transaction. See the discussion on cancelling transactions in *BRM Developer's Guide*.
PCM_OP_TRANS_POL_COMMIT	Recommended. Enables you to customize how to commit an open transaction. See the discussion on committing transactions in *BRM Developer's Guide*.
PCM_OP_TRANS_POL_OPEN	Recommended. Enables you to customize how to open a transaction. See the discussion on using transactions in *BRM Developer's Guide*.

Table 1-10 (Cont.) Base Opcodes

Base Opcodes	Description
PCM_OP_TRANS_POL_PREP_COMMIT	Recommended. Enables you to customize how to verify that a transaction can be committed. See the discussion on customizing how to verify the readiness of an external system to commit a transaction opcode in *BRM Developer's Guide*.
PCM_OP_WRITE_FLDS	Recommended. Writes fields in an object. See the discussion on writing fields in objects in *BRM Developer's Guide*.

Batch Suspense Manager FM Standard Opcodes

The batch suspense manager standard opcodes manage batch files for suspended event records stored in the BRM database as **/suspended_batch** objects.

For information about suspending and recycling event records, see *ECE Implementing Charging*.

Include the **ops/batch_suspense.h** header file in all applications that call these opcodes.

Table 1-11 Batch Suspense Manager FM Standard Opcodes

Opcode	Description
PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES	Recommended. Deletes suspended batches from the BRM database. See "Deleting Records for Suspended Batches".
PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES	Recommended. Resubmits the batches that have been suspended. See "Resubmitting Suspended Batches".
PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES	Recommended. Writes off the batches which are at the “Suspended" stage because of some business rule. See "Writing Off Suspended Batches".

Billing FM Policy Opcodes

Use the billing policy opcodes to customize billing and A/R processes.

Include the **ops/bill.h** header file in all applications that call these opcodes.

Table 1-12 Billing FM Policy Opcodes

Opcode	Description
PCM_OP_BILL_POL_BILL_PRE_COMMIT	Recommended. Enables you to perform custom modifications to a bill object before it is committed to the BRM database. See "Modifying a Bill Object".
PCM_OP_BILL_POL_CALC_PYMT_DUE_T	Recommended. Enables you to customize how to calculate the due date and the payment collection date of a bill (**/bill** object). See the following topics: • Calculating When a Bill is Due • Payment Due Dates for Corrective Bills • Calculating Payment Collection Dates
PCM_OP_BILL_POL_CHECK_SUPPRESSION	Recommended. Enables you to customize exceptions to bill suppressions. See "Determining Whether Bills Should Be Suppressed" and "Suppressing Bills".
PCM_OP_BILL_POL_EVENT_SEARCH	Recommended. Enables you to customize how to search for events associated with an account. See "Finding Events Associated with an Account".
PCM_OP_BILL_POL_GET_EVENT_SPECIFIC_DETAILS	Recommended. Enables you to customize how to get event specific details based on the type of the event. This opcode is called by PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT.
PCM_OP_BILL_POL_GET_ITEM_TAG	Recommended. Enables you to customize the event attributes that set and return a customized item tag. See the discussion on creating custom bill items in *BRM Configuring and Running Billing*.
PCM_OP_BILL_POL_GET_PENDING_ITEMS	Recommended. Enables you to customize how to select the pending items to be included in a bill. See "Customizing Bill Now"
PCM_OP_BILL_POL_POST_BILLING	Recommended. Enables you to perform custom processing on a bill unit at the time of billing. See "About Suspending Billing of Accounts and Bills"
PCM_OP_BILL_POL_POST_SET_LIMIT_AND_CR	Recommended. Enables you to customize the notification sent after a temporary credit limit is created or modified. See "Managing Credit Limits and Sub-Balance Consumption Rules".
PCM_OP_BILL_POL_PRE_SET_LIMIT_AND_CR	Recommended. Enables you to modify the validity dates for a temporary credit limit. See "Managing Credit Limits and Sub-Balance Consumption Rules".

Table 1-12 (Cont.) Billing FM Policy Opcodes

Opcode	Description
PCM_OP_BILL_POL_REVERSE_PAYMENT	Recommended. Enables you to implement additional rules for performing write-off reversals and allocating funds to written-off accounts and bill units. See "Customizing Reversal of Payments Allocated to Written-Off Accounts".
PCM_OP_BILL_POL_SPEC_BILLNO	Recommended. Enables you to customize bill numbers. See "Customizing the Format of Bill and Invoice Numbers".
PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE	Recommended. Enables you to customize accounting cycles. See "Customizing Accounting Cycles"
PCM_OP_BILL_POL_VALID_ADJUSTMENT	Recommended. Enables you to customize item adjustment validation criteria. See "Customizing Item Adjustments".
PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL	Recommended. Enables you to customize how BRM determines if a corrective bill can be generated. See "Validating Bills for the Corrective Billing Process" and "Policy Bill Validations".
PCM_OP_BILL_POL_VALID_DISPUTE	Recommended. Enables you to customize how to validate fields in an **/item** object when a dispute is made. See "Customizing Item Disputes".
PCM_OP_BILL_POL_VALID_SETTLEMENT	Recommended. Enables you to customize how to validate the information used for settling an item which is in dispute. See "Customizing Item Disputes" and "Settling Disputed Items".
PCM_OP_BILL_POL_VALID_TRANSFER	Recommended. Enables you to customize how to validate the information used for transferring money from the payment item to the target item. See "Customizing Item Transfer Validation".
PCM_OP_BILL_POL_VALID_WRITEOFF	Recommended. Enables you to customize how items are validated for a write-off. See "Customizing Write-Off Validation".

Billing FM Standard Opcodes

The billing standard opcodes manage billing and billing group processes, as well as some A/R and payment processes.

To customize groups, use the billing opcodes, such as PCM_OP_BILL_GROUP_CREATE. The billing opcodes call the group opcodes.

Include the **ops/bill.h** header file in all applications that call these opcodes.

Table 1-13 Billing FM Standard Opcodes

Opcode	Description
PCM_OP_BILL_CREATE_SPONSORED_ITEMS	Recommended. Creates **/item/sponsor** objects for sponsoring accounts. See "Creating /item/sponsor Objects".
PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS	Recommended. Converts currency amounts from a source currency to a destination currency. See "About Currency Conversion".
PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES	Recommended. Supplies currency conversion rates. See "About Currency Conversion".
PCM_OP_BILL_CYCLE_TAX	Last Resort. Calculates tax on deferred taxable amounts. See "Calculating Taxes during Billing".
PCM_OP_BILL_DEBIT	Recommended. Debits or credits a noncurrency balance. See "Applying Debits and Credits".
PCM_OP_BILL_FIND	Recommended. Locates a **/bill** object, given a bill number. See "Finding a Bill".
PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT	Limited. Retrieves the discount for events of a given bill item. See "Finding Discounts in Bill Items".
PCM_OP_BILL_GET_LIMIT	Recommended. Retrieves the permanent and temporary credit limits for an account. See "Retrieving Credit Limits".
PCM_OP_BILL_GROUP_ADD_MEMBER	Last Resort. Adds one or more accounts to an existing account group. See "Adding Accounts to Account Hierarchies".
PCM_OP_BILL_GROUP_CREATE	Last Resort. Creates a new account group (**/group** object.) for billing purposes. See "Creating Account Hierarchies".
PCM_OP_BILL_GROUP_DELETE	Last Resort. Deletes an existing **/group** object. See "Deleting Accounts from Account Hierarchies".
PCM_OP_BILL_GROUP_DELETE_MEMBER	Last Resort. Deletes an account from an existing group. See "Deleting Accounts from Account Hierarchies".
PCM_OP_BILL_GROUP_GET_CHILDREN	Last Resort. Gets child accounts of a specified account group. See "Getting a List of Child Accounts in an Account Hierarchy".

Table 1-13 (Cont.) Billing FM Standard Opcodes

Opcode	Description
PCM_OP_BILL_GROUP_GET_PARENT	Last Resort. Gets the parent account of a given account group. See "Finding the Parent of an Account Hierarchy".
PCM_OP_BILL_GROUP_MOVE_MEMBER	Recommended. Moves a group member; deletes the group if it is empty, and creates the new group if it does not exist. See "Moving Accounts from One Account Hierarchy to Another".
PCM_OP_BILL_ITEM_EVENT_SEARCH	Recommended. Searches the **/event** object for details related to a specific item. See "Finding Events Associated with Bill Items".
PCM_OP_BILL_ITEM_REFUND	Recommended. Creates a refund item for a bill or bill unit. See "Refunding Payments".
PCM_OP_BILL_ITEM_TRANSFER	Limited. Transfers money from a source item to a target item. See "Transferring Balances between Items".
PCM_OP_BILL_MAKE_BILL	Last Resort. Creates a bill for a specified bill unit See "Creating Bills".
PCM_OP_BILL_MAKE_BILL_NOW	Last Resort. Bills a specified bill unit immediately. See "Running Bill Now".
PCM_OP_BILL_MAKE_BILL_ON_DEMAND	Last Resort. Creates a bill immediately after a bill unit is created or when a bundle is purchased. See "On-Purchase Billing".
PCM_OP_BILL_MAKE_CORRECTIVE_BILL	Recommended. Creates a corrective bill for a bill unit. See "Making a Corrective Bill".
PCM_OP_BILL_MAKE_TRIAL_BILL	Limited. Creates a trial invoice and collects revenue assurance data for trial billing. See "Making Trial Bills".
PCM_OP_BILL_RCV_PAYMENT	Limited. Creates a payment item and records that currency has been received. See "Receiving Payments".
PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION	Recommended. Deactivates manual account suppression immediately or on a specified future date. See "Ending Manual Account Suppression".
PCM_OP_BILL_REVERSE	Last Resort. Opens a closed payment item and removes the credit. This opcode is a wrapper for PCM_OP_BILL_REVERSE_PAYMENT. See "Reversing Payments".

Table 1-13 (Cont.) Billing FM Standard Opcodes

Opcode	Description
PCM_OP_BILL_REVERSE_PAYMENT	Limited. Reverses a payment. See "Reversing Payments".
PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION	Recommended. Activates manual account suppression immediately or on a specified future date. See "Suppressing Accounts".
PCM_OP_BILL_SET_BILL_SUPPRESSION	Recommended. Handles manual bill suppression. See "Manually Suppressing Bills".
PCM_OP_BILL_SET_LIMIT_AND_CR	Recommended. Sets a permanent credit limit, a temporary credit limit, and/or a consumption rule for both currency and noncurrency sub-balances. See "Customizing Credit Limit and Consumption Rules".
PCM_OP_BILL_TRANSFER_BALANCE	Recommended. Transfers balances from one balance group to another balance group. See "Transferring Balances between Balance Groups".
PCM_OP_BILL_VIEW_INVOICE	Recommended. Retrieves a formatted invoice from the database. This opcode is deprecated and included only for backward compatibility. Use PCM_OP_INV_VIEW_INVOICE instead.

Channel FM Standard Opcodes

The channel standard opcodes propagate object changes from BRM to a directory server.

See the discussion about the channel framework in *BRM LDAP Manager*.

Include the **ops/channel.h** header file in all applications that call these opcodes.

Table 1-14 Channel FM Standard Opcodes

Opcode	Description
PCM_OP_CHANNEL_PUSH	Recommended. Creates **/channel_event** objects whenever a change occurs to a specified **/account** or **/service** object. You specify which events trigger the opcode to create **/channel_event** objects by using event notification.
PCM_OP_CHANNEL_SYNC	Limited Propagates **/channel_event** objects to the LDAP Server. BRM does not use this opcode, but you can use it for custom applications.

Collections Manager FM Policy Opcodes

Use the collections policy opcodes to customize collections features.

For information about Collections Manager, see "Understanding Collections Manager" in *BRM Collections Manager*.

Include the **ops/collections.h** header file in all applications that call these opcodes.

Table 1-15 Collections Manager FM Policy Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES	Recommended. Enables you to customize how a finance charge is calculated or add collections functionality. See "Applying Finance Charges".
PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES	Recommended. Enables you to customize how the late fee is calculated or to add functionality. See "Applying Late Fees".
PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT	Recommended. Enables you to customize how account bill units are assigned to collections agents. See "Assigning Bill Units Automatically".
PCM_OP_COLLECTIONS_POL_ASSIGN_DCA	Recommended. Enables you to automate the logic of selecting a debt collections agent (DCA) when multiple DCAs are configured. See "Assigning Bill Units to a Debt Collections Agency".
PCM_OP_COLLECTIONS_POL_CALC_DUE_DATE	Recommended. Enables you to customize how due dates are set. "Customizing Due Dates".
PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION	Recommended. Enables your custom application to perform custom collections actions. See "Performing Custom Collections Actions".
PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO	Recommended. Enables your custom application to perform cleanup or other tasks when a bill unit leaves a collections scenario. See "Performing Custom Actions when a Bill Unit Leaves Collections".
PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS	Recommended. Enables you to customize how to assign bill units to collection actions. See "Customizing How to Apply Collections Actions to Collections Group Members".
PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS	Recommended. Enables you to customize how to validate if a scenario is valid for a bill unit. See "Getting All Valid Collections Scenarios".
PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY	Recommended. Enables you to break a promise-to-pay agreement when a customer misses an installment payment. See "Breaking a Promise-to-Pay Agreement".

Table 1-15 (Cont.) Collections Manager FM Policy Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT	Recommended. Enables you to customize auto-collect of the due amount for **promise_to_pay** and **collect_payment** actions, if the customer has opted to pay off the due amount through credit card or direct debit. See "Executing Pending Actions for a Bill Unit".
PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY	Recommended. Enables you to perform additional validation before invoking promise obligation for a bill unit. See "Creating a System-Configured Promise-to-Pay Agreement" and "Creating a Manually Configured Promise-to-Pay Agreement".
PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA	Recommended. Enables you to customize dunning letter data. See "Customizing Dunning Letters".
PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO	Recommended. Enables you to include additional criteria to determine whether bill units should enter and exit collections. **Important:** The source code for this opcode is not shipped with BRM. See "Executing Automatic Collections Actions".
PCM_OP_COLLECTIONS_POL_SELECT_PROFILE	Recommended. Enables you to group bill units into collections profiles based on any criteria you choose. See "Mapping Bill Units to Collections Profiles".

Collections Manager FM Standard Opcodes

The collections standard opcodes identify account bill units with overdue balances and manage activities to collect those balances.

For information about Collections Manager, see "Understanding Collections Manager" in *BRM Collections Manager*.

Include the **ops/collections.h** header file in all applications that call these opcodes.

Table 1-16 Collections Manager FM Standard Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_ADD_ACTION	Recommended. Adds collections actions to a collections scenario of a bill unit. See "Adding Actions to a Collections Scenario".
PCM_OP_COLLECTIONS_ASSIGN_AGENT	Recommended. Assigns bill units to a collections agent. See "Assigning Bill Units to a Collections Agent".
PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS	Recommended. Gets aging bucket details for a bill unit. See "Retrieving Aging Buckets Information".

Table 1-16 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION	Recommended. Deletes a collections action. See "Deleting an Existing Collections Action".
PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE	Recommended. Deletes a collections profile. See "Deleting an Existing Collections Profile".
PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO	Recommended. Deletes an existing collections scenario. See "Deleting an Existing Collections Scenario".
PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS	Recommended. Gets a list of all currently defined collections actions. See "Getting All Currently Defined Collections Actions".
PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES	Recommended. Retrieves a list of currently defined collections profiles. See "Getting All Currently Defined Collections Profiles".
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS	Recommended. Gets a list of all collections scenarios and associated profiles. See "Getting All Currently Defined Collections Scenarios".
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL	Recommended. Gets details of a particular collections scenario. See "Getting Details of a Collections Scenario".
PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES	Recommended. Gets a list of templates. See "Getting a List of Message Templates".
PCM_OP_COLLECTIONS_CONFIG_SET_ACTION	Recommended. Creates or updates a collections action. See "Creating or Updating Collections Actions".
PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE	Recommended. Creates or updates a collections profile. See "Creating or Updating Collections Profiles".
PCM_OP_COLLECTIONS_EXEMPT_BILLINFO	Recommended. Exempts bill units from collections. See "Exempting Bill Units from Collections".
PCM_OP_COLLECTIONS_GET_ACTION_HISTORY	Recommended. Finds historic information about a particular collections action. See "Retrieving Collections Action History Information".
PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS	Recommended. Retrieves a list of collections actions assigned to collections agents. See "Retrieving a List of Collections Actions".
PCM_OP_COLLECTIONS_GET_BILLINFOS	Recommended. Gets a list of bill units that are in collections. See "Retrieving a List of Bill Units in Collections".

Table 1-16 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_GET_DUNNING_LETTER	Recommended. Creates formatted dunning letters. See "Retrieving Dunning Letters".
PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL	Recommended. Gets details of the collections scenario for a bill unit. See "Getting Details of a Collections Scenario".
PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS	Recommended. Determines all the eligible collections scenarios for a bill unit. See "Getting All Valid Collections Scenarios".
PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER	Recommended. Adds members to a collections group. See "Adding Members to a Collections Group".
PCM_OP_COLLECTIONS_GROUP_CREATE	Recommended. Creates the collections group. See "Creating a Collections Group".
PCM_OP_COLLECTIONS_GROUP_DELETE	Recommended. Deletes the collections group. See "Deleting a Collections Group".
PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER	Recommended. Deletes members from the collections group. See "Removing a Member from a Collections Group".
PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO	Recommended. Verifies whether a bill unit belongs to a collections group or not. It also provides information about the pending receivables for each bill unit. If the bill unit is a parent of a collection group, it provides the name of the group and its members. See "Retrieving a Collections Group".
PCM_OP_COLLECTIONS_GROUP_MODIFY	Recommended. This opcode is used to rename the group, change parent, or replace existing members. While replacing the existing members, this opcode validates all the members and replaces the existing set of group members. See "Modifying a Collections Group".
PCM_OP_COLLECTIONS_GROUP_SET_PARENT	Recommended. Sets the parent for a collections group. Validation is performed to verify if the bill unit is already a parent of another collections group. A bill unit can be the owner or parent of only one collections group. See "Changing the Parent of a Collections Group".
PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY	Recommended. For a bill unit that is already in collections, creates a promise-to-pay agreement. See "Creating a System-Configured Promise-to-Pay Agreement" and "Creating a Manually Configured Promise-to-Pay Agreement".

Table 1-16 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description
PCM_OP_COLLECTIONS_PROCESS_BILLINFO	Recommended. Determines whether bill units enter or exit collections and performs collections actions. See "Executing Automatic Collections Actions".
PCM_OP_COLLECTIONS_RESCHEDULE_ACTION	Recommended. Reschedules an action that was scheduled by a bill unit's collections scenario. See "Rescheduling an Action Scheduled for a Bill Unit".
PCM_OP_COLLECTIONS_REPLACE_SCENARIO	Recommended. Replaces the existing collections scenario for a bill unit with a new collections scenario. See "Replacing a Collections Scenario".
PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY	Recommended. Cancels the promise-to-pay agreement, by canceling the outstanding payment milestones, deleting the corresponding schedule objects, rescheduling the scenario actions to start from the next day, and updating the scenario objects. See "Breaking a Promise-to-Pay Agreement".
PCM_OP_COLLECTIONS_SET_ACTION_STATUS	Recommended. Changes the status of a collections action. See "Changing the Status of a Collections Action".
PCM_OP_COLLECTIONS_SET_DUNNING_LETTER	Recommended. Gathers data for system-generated dunning letters. See "Gathering and Storing Data for Dunning Letters".
PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER	Recommended. Prepares an invoice reminder message. See "Preparing Invoice Reminders".
PCM_OP_COLLECTIONS_TAKE_ACTION	Recommended. Runs pending actions for a bill unit. See "Executing Pending Actions for a Bill Unit".
PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS	Recommended. Updates the amount and payment details for **Promise to Pay** and **Collect Payment** actions. See "Updating Amount and Payment Details".

Contract FM Policy Opcodes

Use the contract policy opcodes to customize contracts.

Include the **ops/contract.h** header file in all applications that call these opcodes.

Table 1-17 Contract FM Policy Opcodes

Opcode	Description
PCM_OP_CONTRACT_POL_CANCEL_CONTRACT	Recommended. Enables you to customize how penalty fees are applied during cancellation. By default, the opcode does nothing.
PCM_OP_CONTRACT_POL_POST_CREATE_CONTRACT	Recommended. Enables you to customize how contracts are created. By default, the opcode does nothing.
PCM_OP_CONTRACT_POL_PREP_CONTRACT	Recommended. Enables you to customize how contracts are created. By default, the opcode does nothing.
PCM_OP_CONTRACT_POL_VALID_CONTRACT	Recommended. Validates a customer's contract for valid start and end dates, pricing objects, account objects, and contract objects (during modification). The opcode is triggered during contract creation, modification, and renewal.

Contract FM Standard Opcodes

Use the contract standard opcodes to manage customer contracts.

Include the **ops/contract.h** header file in all applications that call these opcodes.

Table 1-18 Contract FM Policy Opcodes

Opcode	Description
PCM_OP_CONTRACT_CANCEL_CONTRACT	Recommended. Cancels a customer's existing contract and applies an early termination fee if required by the contract terms. See "Canceling Contracts".
PCM_OP_CONTRACT_CREATE_CONTRACT	Recommended. Creates a contract for a customer. See "Creating Contracts".
PCM_OP_CONTRACT_MODIFY_CONTRACT	Recommended. Modifies a customer's existing contract. See "Modifying Contracts".
PCM_OP_CONTRACT_RENEW_CONTRACT	Recommended. Renews a customer's contract. See "Renewing Contracts".

Customer FM Policy Opcodes

Use the customer policy opcodes to customize the business logic to process account information during customer account creation.

Include the **ops/cust.h** header file in all applications that call these opcodes.

Table 1-19 Customer FM Policy Opcodes

Opcode	Description
PCM_OP_CUST_POL_CANONICALIZE	Recommended. Enables you to customize canonicalization when you create a localized version of BRM. See the discussion on creating a localized version of BRM in *BRM Developer's Guide*.
PCM_OP_CUST_POL_COMPARE_PASSWD	Recommended. Enables you to customize the password encryption mechanism. See "Implementing Password Encryption".
PCM_OP_CUST_POL_DECRYPT_PASSWD	Recommended. Enables you to customize how to decrypt a password. See "Implementing Password Encryption".
PCM_OP_CUST_POL_ENCRYPT_PASSWD	Recommended. Enables you to customize password encryption. See "Implementing Password Encryption".
PCM_OP_CUST_POL_EXPIRATION_PASSWD	Recommended. Enables you to customize how passwords are expired. See "Customizing Password Expiration".
PCM_OP_CUST_POL_GET_CONFIG	Recommended. Enables you to customize the configuration data that should be returned to an external customer management system. See "Sending Account Information to Your Application when an Account is Created".
PCM_OP_CUST_POL_GET_DB_LIST	Recommended. Enables you to create and customize a list of schemas to use when creating accounts. See "Getting a List of Database Schemas".
PCM_OP_CUST_POL_GET_DB_NO	Recommended. Enables you to choose which schema to use when creating accounts. See "Selecting a Database Schema"
PCM_OP_CUST_POL_GET_DEALS	Recommended. Enables you to retrieve a customized list of bundles from the BRM database. See "Getting Data about Bundles, Charge Offers, Discount Offers, and Services".
PCM_OP_CUST_POL_GET_INTRO_MSG	Recommended. Enables you to customize how to send an introductory message to a customer. See "Customizing the Introductory Message".
PCM_OP_CUST_POL_GET_PLANS	Recommended. Enables you to retrieve a customized list of packages from the BRM database. See "Getting Data about Bundles, Charge Offers, Discount Offers, and Services".

Table 1-19 (Cont.) Customer FM Policy Opcodes

Opcode	Description
PCM_OP_CUST_POL_GET_POPLIST	Recommended. Enables you to customize how to return a point-of-presence (POP) list. See "Returning a Point-of-Presence (POP) List".
PCM_OP_CUST_POL_GET_PRODUCTS	Recommended. Enables you to retrieve a customized list of charge offers from the BRM database. See "Getting Data about Bundles, Charge Offers, Discount Offers, and Services".
PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS	Recommended. Enables you to retrieve a customized list of packages, bundles, or both that an account owns. See "Getting a List of Packages and Bundles That an Account Owns".
PCM_OP_CUST_POL_POST_COMMIT	Recommended. Enables you to customize interactions with an external account creation application before an account is committed during customer account creation. See: • Adding Custom Account Creation Steps After the Account Is Committed • Creating Accounts • Customizing the Introductory Message
PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER	Recommended. Enables you to send information to an external account creation application when an account is modified. See "Sending Account Information to Your Application when an Account is Modified".
PCM_OP_CUST_POL_PRE_COMMIT	Recommended. Enables you to customize interactions with an external account creation application after an account is committed during customer account creation. See "Adding Custom Account Creation Steps Before the Account Is Committed".
PCM_OP_CUST_POL_PRE_DELETE_PAYINFO	Recommended. Enables you to add custom actions before a **/payinfo** object is deleted. See "Customizing Customer Payment Information".
PCM_OP_CUST_POL_PREP_AACINFO	Recommended. Enables you to prepare automatic account creation (AAC) data for validation. See "Customizing Automatic Account Creation (AAC) Information".
PCM_OP_CUST_POL_PREP_ACCTINFO	Recommended. Enables you to customize account numbers. See "Customizing Account Numbers".

Table 1-19 (Cont.) Customer FM Policy Opcodes

Opcode	Description
PCM_OP_CUST_POL_PREP_BILLINFO	Recommended. Enables you to customize the creation of a **/billinfo** object during customer account creation or while updating billing information to prepare for validation. See "Preparing Bill Unit Data".
PCM_OP_CUST_POL_PREP_INHERITED	Recommended. Enables you to customize inherited information when a service object is created. See "Creating Services".
PCM_OP_CUST_POL_PREP_LIMIT	Recommended. Enables you to customize credit limit information prior to validation. See "Managing Credit Limits and Sub-Balance Consumption Rules".
PCM_OP_CUST_POL_PREP_LOCALE	Recommended. Enables you to customize how to set the locale for an account. See "Customizing Locale Information".
PCM_OP_CUST_POL_PREP_LOGIN	Recommended. Enables you to customize login names for services. See "Customizing Login Names".
PCM_OP_CUST_POL_PREP_NAMEINFO	Recommended. Enables you to customize how name and address information is processed. See "Customizing Name and Address Information".
PCM_OP_CUST_POL_PREP_PASSWD	Recommended. Enables you to customize password creation. See "Customizing Passwords".
PCM_OP_CUST_POL_PREP_PAYINFO	Limited Enables you to customize how **/payinfo** objects are created. See "Customizing Payment Method Data Preparation".
PCM_OP_CUST_POL_PREP_PROFILE	Limited Enables you to customize how profiles are created. See "Managing and Customizing Profiles".
PCM_OP_CUST_POL_PREP_STATUS	Limited Enables you to customize status data before modifying the status. See "Customizing Status Changes".
PCM_OP_CUST_POL_PREP_TOPUP	Recommended. Enables you to customize how to set up and modified. See "Preparing an Account's Top-Up Information".
PCM_OP_CUST_POL_READ_PLAN	Recommended. Enables you to retrieve a customized list of services, bundles, and charge offers for purchase. See "Getting Packages, Bundles, and Charge Offers for Purchase".

Table 1-19 (Cont.) Customer FM Policy Opcodes

Opcode	Description
PCM_OP_CUST_POL_TAX_CALC	Recommended. Enables you to customize tax calculation. See "Using Custom Tax Rates".
PCM_OP_CUST_POL_TAX_INIT	Recommended. Enables you to customize tax calculation. See "Using Custom Tax Rates".
PCM_OP_CUST_POL_TRANSITION_DEALS	Recommended. Enables you to create and customize a list of bundles and charge offers available for transition. See "Customizing Bundle Transitions".
PCM_OP_CUST_POL_TRANSITION_PLANS	Recommended. Enables you to customize how bundles in a specified package are transitioned. See "Customizing Bundle Transitions". **Note:** To customize how to transition a *package* from one account to another, use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN. See "Customizing Package Transitions without Configuring /transition Objects".
PCM_OP_CUST_POL_VALID_AACINFO	Recommended. Enables you to validate data in the automatic account creation (AAC) fields. See "Customizing Automatic Account Creation (AAC) Information".
PCM_OP_CUST_POL_VALID_ACCTINFO	Recommended. Enables you to customize account numbers. See "Customizing Account Numbers".
PCM_OP_CUST_POL_VALID_BILLINFO	Recommended. Enables you to customize how to validate billing information in a **/billinfo** object. See "Validating Bill Unit Data".
PCM_OP_CUST_POL_VALID_LIMIT	Recommended. Enables you to validate custom credit limit information before it is set in the account. See "Managing Credit Limits and Sub-Balance Consumption Rules".
PCM_OP_CUST_POL_VALID_LOCALE	Recommended. Enables you to customize how to validate locale information before it is set in the account. See "Customizing Locale Information".
PCM_OP_CUST_POL_VALID_LOGIN	Recommended. Enables you to Enables you to customize how to validate login names. See "Customizing Login Names".
PCM_OP_CUST_POL_VALID_NAMEINFO	Recommended. Enables you to customize how customer contact information is validated. See "Customizing Name and Address Information".

Table 1-19 (Cont.) Customer FM Policy Opcodes

Opcode	Description
PCM_OP_CUST_POL_VALID_PASSWD	Recommended. Enables you to customize how passwords are validated. See "Customizing Passwords".
PCM_OP_CUST_POL_VALID_PAYINFO	Limited Enables you to customize how payment methods are validates. See "Customizing Payment Method Validation".
PCM_OP_CUST_POL_VALID_PROFILE	Limited Enables you to customize how profiles are validated. See "Managing and Customizing Profiles".
PCM_OP_CUST_POL_VALID_STATUS	Limited Enables you to customize how status changes are validated. See "Customizing Status Changes".
PCM_OP_CUST_POL_VALID_TAXINFO	Recommended. Enables you to customize how to validate the VAT certificate number provided during account creation. See "Validating Tax Information".
PCM_OP_CUST_POL_VALID_TOPUP	Recommended. Enables you to customize how top-up data is validated. See "Validating an Account's Top-Up Information".

Customer FM Standard Opcodes

The customer standard opcodes manage the creation, deletion, and modification of account information during customer account creation.

Include the **ops/cust.h** header file in all applications that call these opcodes.

> **Important:**
>
> To create an account, use PCM_OP_CUST_COMMIT_CUSTOMER. This wrapper opcode performs all of the tasks necessary to create an active and billable account in the database. The value of its PIN_FLD_TXN_FLAGS input flist field determines how the opcode handles transactions. See "Transaction Handling During Account Creation" for more information.

Table 1-20 Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_ASSOCIATE_ROLE	Recommended. Associates a pcm_client or admin_client service with an existing role. See "Associating a pcm_client or admin_client Service with a Role".

Table 1-20 (Cont.) Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_CHANGE_BUSINESS_PROFILE	Recommended. Changes a bill unit's business profile. See "Changing a Bill Unit's Business Profile".
PCM_OP_CUST_COMMIT_CUSTOMER	Recommended. Creates an account. This opcode is a wrapper opcode that performs all the tasks necessary to create an account. See "Creating Accounts".
PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE	Recommended. Creates one **/associated_bus_profile** object for each bill unit in the account. See "Creating Business Profiles".
PCM_OP_CUST_CREATE_BAL_GRP	Limited Creates an active **/balance_group** object. Use PCM_OP_CUST_SET_BAL_GRP to create the **/balance_group** object. See "Creating Balance Groups ".
PCM_OP_CUST_CREATE_BILLINFO	Limited Creates an active bill unit. See "Creating Bill Units".
PCM_OP_CUST_CREATE_CUSTOMER	Limited Creates an active customer account, including creating and initializing an **/account** object and one or more **/service** objects. See "Creating Accounts".
PCM_OP_CUST_CREATE_PAYINFO	Limited Creates an active **/payinfo** object. See "Creating Accounts".
PCM_OP_CUST_CREATE_PROFILE	Limited Creates a **/profile** object. See "Managing and Customizing Profiles".
PCM_OP_CUST_CREATE_ROLE	Recommended. Creates a **/config/role** object. See "Creating a Role".
PCM_OP_CUST_CREATE_SERVICE	Last Resort Creates a service object with inheritance pass through. See "Creating Services".
PCM_OP_CUST_CREATE_TOPUP	Limited Creates **/topup** and **/group/topup** objects. See "Setting Up Top-Up Information in an Account".
PCM_OP_CUST_DELETE_ACCT	Last Resort Deletes the specified **/account** object and all related objects (such as events, bill items, bill history, balances, and notes) and disassociates devices that are assigned to the services. See "Deleting Accounts".

Table 1-20 (Cont.) Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_DELETE_BAL_GRP	Last Resort Deletes a **/balance_group** object. See "Deleting a Balance Group".
PCM_OP_CUST_DELETE_BILLINFO	Last Resort Deletes the specified bill unit and the balance groups associated with it. See "Deleting Bill Units".
PCM_OP_CUST_DELETE_PAYINFO	Recommended. Deletes the specified **/payinfo** object. See "Customizing Customer Payment Information".
PCM_OP_CUST_DELETE_PROFILE	Limited Deletes the specified **/profile** object. See "Managing and Customizing Profiles".
PCM_OP_CUST_DELETE_ROLE	Recommended. Deletes a **/config/role** object. See "Deleting a Role".
PCM_OP_CUST_DELETE_TOPUP	Limited Deletes **/topup** objects. See "Deleting Top-Ups".
PCM_OP_CUST_FIND	Recommended. Searches for information in an **/account** object given an account number. See "Finding Accounts".
PCM_OP_CUST_FIND_PAYINFO	Recommended. Finds **/payinfo** objects that belong to a specified account. See "Finding Payment Info".
PCM_OP_CUST_FIND_PROFILE	Recommended. Retrieves the **/profile** objects associated with a specified account. See "Retrieving Account Profile Information".
PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO	Recommended. Gets the value of a key in the business profile and validation templates associated with an object. See "Getting Information about a Business Profile".
PCM_OP_CUST_GET_LIFECYCLE_STATES	Recommended. Gets information from the life cycle states configuration object (**/config/lifecycle_states**) associated with a specified service type or gets the configuration object itself. See "Getting Life Cycle States".
PCM_OP_CUST_GET_NEWSFEED	Recommended. Retrieves and returns the news feed information stored in the **/newsfeed** object. See *BRM Managing Customers*.

Table 1-20 (Cont.) Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_GET_NOTE	Recommended. Gets notes and note exchanges based on any of the **/note** object fields, such as the date, the type of note, the status, the CSR who wrote the note, and so on.
PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES	Recommended. Retrieves the subscriber's preferences from the **/profile/subscriber_preferences** object for a subscriber's service or account. See "Maintaining Subscriber's Charging Preferences Data".
PCM_OP_CUST_INIT_SERVICE	Last Resort Initializes a service object. See "Maintaining Subscriber's Charging Preferences Data".
PCM_OP_CUST_MODIFY_BAL_GRP	Limited Modifies the specified **/balance_group** object. See "Creating Balance Groups ".
PCM_OP_CUST_MODIFY_CUSTOMER	Recommended. Modifies customer account information. See "Modifying an Account" and "Moving a Balance Group from One Bill Unit to Another".
PCM_OP_CUST_MODIFY_PAYINFO	Limited Modifies selected fields in the **/payinfo** object. See "Customizing Customer Payment Information".
PCM_OP_CUST_MODIFY_PROFILE	Limited Modifies the specified **/profile** object. See "Managing and Customizing Profiles".
PCM_OP_CUST_MODIFY_SERVICE	Limited Modifies the specified **/service** object. See "Modifying Services".
PCM_OP_CUST_MODIFY_TOPUP	Limited Modifies **/topup** and **/group/topup** objects. See "Setting Up Top-Up Information in an Account".
PCM_OP_CUST_PREP_CUSTOMER	Recommended. Validates customer information prior to account creation. See "Creating Accounts".
PCM_OP_CUST_SET_ACCTINFO	Limited Initializes an **/account** object with generic fields passed in on the input flist. See "Customizing Account Numbers".
PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE	Recommended. Updates the **/associated_bus_profile** objects whenever invoice business profiles are modified in the **/config/business_profile** object. See "Assigning Bill Units to Business Profiles".

Table 1-20 (Cont.) Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_SET_BAL_GRP	Limited Performs all necessary tasks to set up the **/balance_group** object and create a link to the customer account. See "Creating Balance Groups ".
PCM_OP_CUST_SET_BILLINFO	Limited Updates billing information in a bill unit for a specified account. See "Updating Bill Units".
PCM_OP_CUST_SET_LOCALE	Limited Sets the Locale field of a specified **/account** object. See "Customizing Locale Information".
PCM_OP_CUST_SET_LOGIN	Limited Updates the service login. See "Customizing Login Names".
PCM_OP_CUST_SET_NAMEINFO	Limited Sets account contact information such as customer name, address, and phone number. See "Managing Name and Address Information".
PCM_OP_CUST_SET_NOTE	Recommended. Creates a **/note** object or modifies an existing one. If you pass in a type-only POID, a **/note** object is created. Otherwise, this opcode modifies the **/note** object. This opcode generates the following events: • **/event/customer/note/create** • **/event/customer/note/modify**
PCM_OP_CUST_SET_PASSWD	Limited Updates the service password for a customer. See "Creating Passwords".
PCM_OP_CUST_SET_PAYINFO	Limited Adds or updates the payment information for a bill unit. See "Customizing Customer Payment Information".
PCM_OP_CUST_SET_STATUS	Limited Sets the status of an **/account**, **/billinfo**, or **/service** object. See "Changing the Status of an Account, Bill Unit, or Service".
PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES	Recommended. Creates, modifies, or deletes a **/profile/subscriber_preferences** object for a specified service or account. See "Maintaining Subscriber's Charging Preferences Data".
PCM_OP_CUST_SET_TAXINFO	Recommended. Adds or updates the tax information in the account object. See "Adding Tax Information to Accounts".
PCM_OP_CUST_SET_TOPUP	Recommended. Sets up standard top-ups and sponsored top-ups. See "Setting Up Top-Up Information in an Account".

Table 1-20 (Cont.) Customer FM Standard Opcodes

Opcode	Description
PCM_OP_CUST_UPDATE_CUSTOMER	Recommended. Updates several pieces of customer information in one operation. See "Modifying an Account".
PCM_OP_CUST_UPDATE_ROLE	Recommended. Updates a **/config/role** object to add opcodes that can be run and storable classes that can be updated by users with that role. See "Modifying a Role".
PCM_OP_CUST_UPDATE_SERVICES	Recommended. Modifies service information for multiple services in one operation. See "Modifying Services".
PCM_OP_CUST_VALID_FLD	Recommended. Validates fields on the input flist based on the information contained in the **/config/fld_validate** object.
PCM_OP_CUST_VALIDATE_CUSTOMER	Recommended. Validates customer information during account creation. This opcode is called by several different opcodes when account information is created or changed. See "Customer Management Opcode Workflows".

Customer Care FM Standard Opcodes

Use the customer care standard opcodes to manage customers. These opcodes are used by the Billing Care and Customer Center applications.

Include the **ops/custcare.h** header file in all applications that call these opcodes.

Table 1-21 Device FM Policy Opcodes

Opcode	Description
PCM_OP_CUSTCARE_MOVE_ACCT	Recommended. Moves an account into or out of an account hierarchy.

Deposit FM Policy Opcodes

Use the deposit policy opcodes to customize deposits.

See "About Deposits" in *BRM Managing Customers*.

Include the **ops/deposit.h** header file in all applications that call these opcodes.

Table 1-22 Deposit FM Policy Opcodes

Opcode	Description
PCM_OP_DEPOSIT_POL_POST_ADD_INTEREST	Recommended. Enables you to add information to the calculated interest amount and interest period. See "Customizing the Interest Calculation".
PCM_OP_DEPOSIT_POL_PRE_ADD_INTEREST	Recommended. Enables you to update the **EFFECTIVE_T** for interest calculation. Any changes to the **EFFECTIVE_T** from this opcode impacts the interest calculation such that the interest is applied on per-day basis. See "Customizing the Interest Calculation".
PCM_OP_DEPOSIT_POL_VALID_COLLECT_PAYMENT	Recommended. Enables you to validate the PCM_OP_DEPOSIT_COLLECT_PAYMENT opcode. See "Validating Deposit Payment Collections".
PCM_OP_DEPOSIT_POL_VALID_PURCHASE_DEPOSIT	Recommended. Enables you to validate the deposit purchase. See "Validating a Deposit Purchase".
PCM_OP_DEPOSIT_POL_VALID_REFUND_REQUEST	Recommended. Enables you to validate the deposit refund request. See "Validating a Deposit Refund Request".
PCM_OP_DEPOSIT_POL_VALID_RELEASE_DEPOSIT	Recommended. Enables you to validate the deposit release. See "Validating a Deposit Release".
PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT	Recommended. Enables you to validate a deposit reversal. See "Validating a Deposit Reversal".
PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION	Recommended. Enables you to validate the **/deposit_specification** object. See "Validating Deposit Specifications".
PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE	Recommended. Enables you to validate the input flist for PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE opcode. See "Validating Deposit Specification Profiles".
PCM_OP_DEPOSIT_POL_VALID_TRANSFER_DEPOSIT	Recommended. Enables you to validate a deposit transfer. See "Validating a Deposit Transfer".
PCM_OP_DEPOSIT_POL_VALID_UPDATE_DEPOSIT	Recommended. Enables you to validate a request to update a deposit with PCM_OP_DEPOSIT_UPDATE_DEPOSIT. See "Validating Deposit Updates".

Deposit FM Standard Opcodes

The deposit opcodes manage customer deposits.

See "About Deposits" in *BRM Managing Customers*.

Include the **ops/deposit.h** header file in all applications that call these opcodes.

Table 1-23 Deposit FM Standard Opcodes

Opcode	Description
PCM_OP_DEPOSIT_ADD_INTEREST	Limited. Calculates the interest amount for a given purchased deposit and period. Used during deposit release, reversal, transfer, update or refund, and called by the **pin_deposit_calc_interest** utility. See "Adding Interest to a Deposit".
PCM_OP_DEPOSIT_COLLECT_PAYMENT	Recommended. Makes the up-front payment for a deposit. See "Collecting Deposit Payments".
PCM_OP_DEPOSIT_CREATE_SPECIFICATION	Recommended. Creates the **/deposit_specification** object. See "Creating Deposit Specifications".
PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE	Recommended. Creates the **/config/deposit_spec_profile** object. See "Creating Deposit Specification Profiles".
PCM_OP_DEPOSIT_DELETE_DRAFT	Recommended. Deletes a **/deposit_specification** object that is in draft state. See "Deleting Draft Deposit Specifications".
PCM_OP_DEPOSIT_GET_DEPOSITS	Limited. Get the purchased deposits of the given account, service, or bill unit from the **/purchased_deposit** object. See "Getting Deposits".
PCM_OP_DEPOSIT_GET_REFUND_REQUEST	Recommended. Gets all deposit refund requests. See "Getting Refund Requests".
PCM_OP_DEPOSIT_GET_SPECIFICATION	Recommended. Gets the **/deposit_specification** object. See "Getting Deposit Specifications".
PCM_OP_DEPOSIT_GET_SPECIFICATION_PROFILE	Recommended. Gets the **/config/deposit_spec_profile** objects. See "Getting Deposit Specification Profiles".

Table 1-23 (Cont.) Deposit FM Standard Opcodes

Opcode	Description
PCM_OP_DEPOSIT_GET_TRANSACTIONS	Recommended. Gets the transaction details of a purchased deposit. It returns the complete details of the events. See "Retrieving Transaction Details of a Deposit".
PCM_OP_DEPOSIT_MODIFY_SPECIFICATION	Recommended. Modifies the **/deposit_specification** object. See "Modifying Deposit Specifications".
PCM_OP_DEPOSIT_MODIFY_SPECIFICATION_PROFILE	Recommended. Updates the **/config/deposit_spec_profile** object. See "Modifying Deposit Specification Profiles".
PCM_OP_DEPOSIT_PURCHASE_DEPOSIT	Recommended. Creates a deposit for a given customer. This opcode triggers a deposit charge event and deposit charge item, creates the **/purchased_deposit** object, and makes the up-front payment for the deposit. See "Purchasing Deposits".
PCM_OP_DEPOSIT_REFUND_REQUEST	Recommended. Creates a deposit refund request. See "Requesting Deposit Refunds".
PCM_OP_DEPOSIT_RELEASE_DEPOSIT	Recommended. Releases an existing deposit as follows, depending on the request type: • Prepayment : Creates **/item/payment** immediately. • Refund Request: Creates **/deposit_refund_request**, which can then be approved if needed. Once approved, the opcode creates **/item/payment** and **/item/refund**. • Zeroize: Settles dues and raises a refund request for the remaining amount. Creates **/item/payment** immediately, makes a call to PCM_OP_BILL_MAKE_BILL_NOW to bill the pending items, allocates the **/item/payment** to the bill items, and refunds the remaining balance to the customer. See "Releasing Deposits".
PCM_OP_DEPOSIT_REVERSE_DEPOSIT	Recommended. Reverses deposits by opening the **/item/payment** made against **/item/deposit_charge**. See "Reversing Deposits".
PCM_OP_DEPOSIT_TRANSFER_DEPOSIT	Recommended. Transfers deposit from one account to another account. See "Transferring Deposits".

Table 1-23 (Cont.) Deposit FM Standard Opcodes

Opcode	Description
PCM_OP_DEPOSIT_TRIGGER_DEPOSIT	Limited. Creates a **/purchased_deposit** object when any charge offers with event type of **/event/billing/product/fee/purchase/deposit** are created as part of a deal purchase. See "Triggering Deposits".
PCM_OP_DEPOSIT_UPDATE_DEPOSIT	Recommended. Increases or decreases the amount of an already purchased deposit. This opcode reverses the existing deposit charge item, creates a new deposit charge for the given amount, and updates the same purchased deposit object. See "Updating Deposits".
PCM_OP_DEPOSIT_UPDATE_DEPOSIT_RECEIVED	Recommended. Called through event notification for **/event/billing/item/transfer** to update the received amount and balance amount, and to map **/event/billing/item/transfer** to this PIN_FLD_EVENTS array. See "Updating Received Deposits".
PCM_OP_DEPOSIT_UPDATE_REFUND_REQUEST	Recommended. Updates the deposit refund review request. See "Updating Refund Requests".

Device FM Policy Opcodes

Use the device policy opcodes to customize device management.

See the discussion about device management in *BRM Developer's Guide*.

Include the **ops/device.h** header file in all applications that call these opcodes.

Table 1-24 Device FM Policy Opcodes

Opcode	Description
PCM_OP_DEVICE_POL_ASSOCIATE	Recommended. Enables you to customize validation during device-to-service association and disassociation. See "Associating /service and /device Objects".
PCM_OP_DEVICE_POL_CREATE	Recommended. Enables you to customize validation during device creation. See "Creating /device Objects".
PCM_OP_DEVICE_POL_DELETE	Recommended. Enables you to customize validation during device deletion. See "Deleting /device Objects".

Table 1-24 (Cont.) Device FM Policy Opcodes

Opcode	Description
PCM_OP_DEVICE_POL_SET_ATTR	Recommended. Enables you to customize validation of device attribute changes. See "Changing the Attributes of /device Objects".
PCM_OP_DEVICE_POL_SET_STATE	Recommended. Enables you to customize validation during device state changes. See "Changing the State of a /device Object".

Device FM Standard Opcodes

The device standard opcodes run device management processes.

See the discussion about device management in *BRM Developer's Guide*.

Include the **ops/device.h** header file in all applications that call these opcodes.

Table 1-25 Device FM Standard Opcodes

Opcode	Description
PCM_OP_DEVICE_ASSOCIATE	Recommended. Associates or disassociates **/service** objects with /device objects. See "Associating /service and /device Objects".
PCM_OP_DEVICE_CREATE	Recommended. Creates a **/device** object of the type specified in the input flist. See "Creating /device Objects".
PCM_OP_DEVICE_DELETE	Recommended. Deletes a **/device** object. See "Deleting /device Objects".
PCM_OP_DEVICE_SET_ATTR	Recommended. Changes the attributes for a **/device** object. See "Changing the Attributes of /device Objects".
PCM_OP_DEVICE_SET_STATE	Recommended. Sets the state for a **/device** object. See "Changing the State of a /device Object".
PCM_OP_DEVICE_UPDATE	Recommended. Changes any combination of attribute values state of a device in a single transaction. See "Changing Device Attributes and States".

Email Data Manager Opcodes

The Email Data Manager opcodes are base opcodes. They provide a different implementation from the standard BRM base opcodes. Unlike FM opcodes, which belong to the Connection Manager, the Email DM opcodes are part of the Email DM.

Include the **ops/base.h** header file in all applications that call these opcodes.

Table 1-26 Email Data Manager Base Opcodes

Email Data Manager base opcode	Description
PCM_OP_CREATE_OBJ	Recommended. Provides a platform-independent interface to **dm_email** for sending one or more email attachments. See "Sending the Introductory Message".
PCM_OP_DELIVERY_MAIL_SENDMSGS	Recommended. Queries the **pin.conf** file for the location of the Email DM, and ensures the data in the PIN_FLD_MESSAGES array is valid. See "Sending the Introductory Message".

General Ledger FM Policy Opcodes

Use the general ledger policy opcodes to customize the data in exported G/L reports.

See *BRM Collecting General Ledger Data* for more information.

Include the **ops/gl.h** header file in all applications that call these opcodes.

Table 1-27 General Ledger FM Policy Opcodes

Opcode	Description
PCM_OP_GL_POL_EXPORT_GL	Recommended. Enables you to customize data in the exported G/L reports. See "Customizing G/L Reports for Export".
PCM_OP_GL_POL_PRE_UPDATE_JOURNAL	Recommended. Enables you to customize general ledger data before it is recorded into **/journal** objects. See "Customizing G/L Data Stored in /journal Objects".

General Ledger FM Standard Opcode

The general ledger standard opcodes calculate account information and create **/ledger_report** objects.

See *BRM Collecting General Ledger Data* for more information.

Include the **ops/gl.h** header file in all applications that call these opcodes.

Table 1-28 General Ledger FM Standard Opcode

Opcode	Description
PCM_OP_GL_LEDGER_REPORT	Recommended. Creates **/ledger_report** objects, which store general ledger reports. . See "How BRM Stores General Ledger Reports".

GPRS Manager 3.0 FM Policy Opcode

Use the GPRS policy opcodes to customize the GPRS service extensions.

See the discussion on GPRS Manager in *BRM Telco Integration*.

Include the **ops/gprs.h** header file in all applications that call these opcodes.

Table 1-29 GPRS Manager 3.0 FM Policy Opcode

Opcode	Description
PCM_OP_GPRS_POL_APPLY_PARAMETER	Recommended. Enables you to customize GPRS service extensions. See "Updating Custom GPRS Service Fields".

GPRS Manager 3.0 FM Standard Opcode

The GPRS standard opcodes add GPRS-specific service extensions.

See the discussion on GPRS Manager in *BRM Telco Integration*.

Include the **ops/gprs.h** header file in all applications that call these opcodes.

Table 1-30 GPRS Manager 3.0 FM Standard Opcode

Opcode	Description
PCM_OP_GPRS_APPLY_PARAMETER	Recommended. Reads the service extensions from the input flist and adds corresponding GPRS service values. See "Updating Custom GPRS Service Fields".

Group FM Standard Opcodes

The group standard opcodes create and delete account groups and account group members.

To customize groups, use the billing opcodes, such as PCM_OP_BILL_GROUP_CREATE. The billing opcodes call the group opcodes.

For information about groups, see "Managing Account and Bill Unit Hierarchies" in *BRM Managing Customers*.

Include the **ops/group.h** header file in all applications that call these opcodes.

Table 1-31 Group FM Standard Opcodes

Opcode	Description
PCM_OP_GROUP_ADD_MEMBER	Limited Adds members to a group. See "Adding Members to Groups".

Table 1-31 (Cont.) Group FM Standard Opcodes

Opcode	Description
PCM_OP_GROUP_CREATE_GROUP	Limited Creates a new group object. See "Creating Groups".
PCM_OP_GROUP_DELETE_GROUP	Limited Deletes an existing **/group** object from the database. See"Deleting Groups" .
PCM_OP_GROUP_DELETE_MEMBER	Limited Deletes members from a group. See "Deleting Members from Groups".
PCM_OP_GROUP_SET_PARENT	Limited Sets the parent object of a group. See "Setting a Group Parent".
PCM_OP_GROUP_UPDATE_INHERITED	Limited Updates the inheritance fields of an existing group. See "Updating Inheritance Fields in Groups".

GSM Manager FM Policy Opcode

Use the GSM Manager policy opcode to add custom service attributes.

See the discussion on GSM Manager in *BRM Telco Integration*.

Include the **ops/gsm.h** header file in all applications that call this opcode.

Table 1-32 GSM Manager FM Policy Opcode

Opcode	Description
PCM_OP_GSM_POL_APPLY_PARAMETER	Recommended. Enables you to customize GSM service extensions. See "Updating Custom GSM Service Fields".

GSM Manager FM Standard Opcode

The GSM Manager standard opcode performs telco provisioning functions.

See the discussion on GSM Manager in *BRM Telco Integration*.

Include the **ops/tcf.h** header file in all applications that call these opcodes.

Table 1-33 GSM Manager FM Standard Opcode

Opcode	Description
PCM_OP_GSM_APPLY_PARAMETER	Recommended. Reads the service extensions from the input flist and adds corresponding GSM service values. See "Updating Custom GSM Service Fields".

Installment FM Policy Opcodes

Use the installment policy opcodes to customize installments.

See "About Installments" in *BRM Configuring and Collecting Payments*.

Include the **ops/installment.h** header file in all applications that call these opcodes.

Table 1-34 Installment FM Policy Opcodes

Opcode	Description
PCM_OP_INSTALLMENT_POL_GET_SCHEDULE_SPEC	Recommended. Enables you to customize getting installment schedule specifications. See "Getting Installment Schedule Specifications".
PCM_OP_INSTALLMENT_POL_PREP_INSTALLMENT	Recommended. Enables you to customize installment fields. See "Preparing Installments".
PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC	Recommended. Enables you to customize installment schedule specification creation fields. See "Preparing Installment Schedule Specifications".
PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT	Recommended. Enables you to validate the information in an installment. See "Validating Installments".
PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC	Recommended. Enables you to validate the installment schedule specification information in the **/config/installment/schedule_spec** object. See "Validating Installment Schedule Specifications".

Installment FM Standard Opcodes

The installment opcodes manage installment processes.

See "About Installments" in *BRM Configuring and Collecting Payments*.

Include the **ops/installment.h** header file in all applications that call these opcodes.

Table 1-35 Installment FM Standard Opcodes

Opcode	Description
PCM_OP_INSTALLMENT_APPLY_CHARGE	Recommended. Applies the installment amount on the effective date. See "Applying Installment Charge".
PCM_OP_INSTALLMENT_CANCEL_INSTALLMENT	Recommended. Cancels the remaining installments and charges the customer for the total amount due. See "Canceling Installments".
PCM_OP_INSTALLMENT_CREATE_INSTALLMENT	Recommended. Create an installment schedule for a customer. See "Creating Installments".
PCM_OP_INSTALLMENT_CREATE_SCHEDULE_SPEC	Recommended. Creates an installment schedule specification, including minimum installment amount, the maximum number of unequal installments, unequal installment frequency, installment type, and eligibility criteria. See "Creating Installment Schedule Specifications".
PCM_OP_INSTALLMENT_GET_INSTALLMENTS	Recommended. Retrieves information about all the installments associated with an account and service. See "Retrieving Installments".
PCM_OP_INSTALLMENT_GET_SCHEDULE_SPEC	Recommended. Retrieves information about the installment schedule specification. See "Retrieving Installment Schedule Specifications".
PCM_OP_INSTALLMENT_MODIFY_SCHEDULE_SPEC	Recommended. Modifies an existing installment schedule specification object. See "Modifying Installment Schedule Specifications".
PCM_OP_INSTALLMENT_PROPOSAL	Recommended. Prepares an installment proposal based on the installment amount, term (the number of months), and start date. See "Installment Proposal".
PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT	Recommended. Updates the **/installment** object for open installments. See "Updating Installments".
PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT_STATUS	Recommended. Changes the status of an installment after a payment is allocated. See "Updating Installment Status".

Invoicing FM Policy Opcodes

The invoicing standard opcodes generate invoices in different formats.

For more information, see *BRM Designing and Generating Invoices*.

Include the **ops/inv.h** header file in all applications that call these opcodes.

Table 1-36 Invoicing FM Policy Opcodes

Opcode	Description
PCM_OP_INV_POL_FORMAT_INVOICE	Recommended. Specifies the format in which to store invoices. See "How Invoices Are Formatted".
PCM_OP_INV_POL_FORMAT_INVOICE_DOC1	Limited Enables you to customize invoices that use the DOC1 format. See "Customizing the Format for DOC1 Invoices".
PCM_OP_INV_POL_FORMAT_INVOICE_HTML	Limited Enables you to customize HTML invoices. See "Customizing the Format for HTML Invoices".
PCM_OP_INV_POL_FORMAT_INVOICE_XSLT	Limited Enables you to customize invoices that use an XSL style sheet. See "Customizing the Invoice Format by Using an XSL Style Sheet".
PCM_OP_INV_POL_FORMAT_INVOICE_XML	Limited Enables you to customize XML invoices. See "Customizing the Format for XML Invoices".
PCM_OP_INV_POL_FORMAT_VIEW_INVOICE	Recommended. Enables you to generate a custom invoice. See "Displaying an Invoice On Demand".
PCM_OP_INV_POL_POST_MAKE_INVOICE	Recommended. Enables you to capture and return errors that were created due to custom invoicing operations defined in PCM_OP_INV_POL_SELECT. See "Configuring Error Checking for Customized Invoicing".
PCM_OP_INV_POL_PREP_INVOICE	Recommended. Enables you to customize invoice formatting prior to formatting and storing. See "Adding Soft Descriptors to Invoices".
PCM_OP_INV_POL_SELECT	Recommended. Enables you to write custom search templates for items and events to be displayed on invoices. See "Customizing Invoice Search Operations" and "Configuring Error Checking for Customized Invoicing".

Invoicing FM Standard Opcodes

The invoicing standard opcodes create and format invoices.

For more information, see *BRM Designing and Generating Invoices*.

Include the **ops/inv.h** header file in all applications that call these opcodes.

Table 1-37 Invoicing FM Standard Opcodes

Opcode	Description
PCM_OP_INV_DECODE_INVOICE_DATA	Recommended. Decodes the value of the PIN_FLD_INVOICE_DATA field in the **/event** object, which contains cached items and events to be displayed on the invoice. See "Decoding Cached Event Data for Invoicing".
PCM_OP_INV_FORMAT_INVOICE	Limited Performs XSL Transformation on an invoice. See "How Invoices Are Formatted".
PCM_OP_INV_MAKE_INVOICE	Recommended. Creates an invoice for a specified regular or corrective bill object. See "How Invoices Are Generated".
PCM_OP_INV_VIEW_INVOICE	Recommended. Retrieves an invoice from the database. See "Displaying Invoices".

IP Address Manager APN FM Policy Opcodes

Use the IP Address Manager APN policy opcodes to perform various checks and maintenance on the APN device.

See the discussion on IP Address Manager in *BRM Telco Integration*.

Include the **ops/apn.h** header file in all applications that call these opcodes.

Table 1-38 IP Address Manager APN FM Policy Opcodes

Opcode	Description
PCM_OP_APN_POL_DEVICE_ASSOCIATE	Recommended. Enables you to customize verification checks on the APN device and the account or service being associated. See "Associating APN with an Account or Service".
PCM_OP_APN_POL_DEVICE_CREATE	Recommended. Enables you to customize verification checks during APN device creation. See "Creating an APN Device".
PCM_OP_APN_POL_DEVICE_DELETE	Recommended. Enables you to customize verification checks during APN device deletion. See "Deleting an APN Device".
PCM_OP_APN_POL_DEVICE_SET_ATTR	Recommended. Enables you to customize how to verify that an APN device is in a state to accept changes. See "Changing the APN Device State".

Table 1-38 (Cont.) IP Address Manager APN FM Policy Opcodes

Opcode	Description
PCM_OP_APN_POL_DEVICE_SET_STATE	Recommended. Enables you to customize how to verify that an APN device is eligible to accept a device state change. See "Changing the APN Device State".

IP Address Manager FM Policy Opcodes

Use the IP Address Manager policy opcodes to perform various checks and maintain the state of the IP address device.

See the discussion on IP Address Manager in *BRM Telco Integration*.

Include the **ops/ip.h** header file in all applications that call these opcodes.

Table 1-39 IP Address Manager FM Policy Opcodes

Opcode	Description
PCM_OP_IP_POL_DEVICE_ASSOCIATE	Recommended. Enables you to customize how to verify that an IP address device is in a state to accept changes. See "Associating an IP Address with Accounts or Services".
PCM_OP_IP_POL_DEVICE_CREATE	Recommended. Enables you to customize verification checks during IP address device creation. See "Creating a Single IP Address Device" and "Creating a Range of IP Address Devices".
PCM_OP_IP_POL_DEVICE_DELETE	Recommended. Enables you to customize verification checks during IP address device deletion. See "Deleting an IP Address Device".
PCM_OP_IP_POL_DEVICE_SET_ATTR	Recommended. Enables you to customize how to verify that an IP address device is in a state to accept changes. See "Modifying an IP Address Device".
PCM_OP_IP_POL_DEVICE_SET_STATE	Recommended. Enables you to customize how to verify that an IP address device is eligible to accept a device state change. See "Changing IP Device States from Unallocated to Returned".

IP Address Manager FM Standard Opcodes

The IP Address Manager standard opcodes create, delete, and maintain the attributes and state of one or more IP address devices.

See the discussion on IP Address Manager in *BRM Telco Integration*.

Include the **ops/ip.h** header file in all applications that call these opcodes.

Table 1-40 IP Address Manager FM Standard Opcodes

Opcode	Description
PCM_OP_IP_DEVICE_CREATE	Recommended. Creates one or more IP address devices. See "Creating a Single IP Address Device" and "Creating a Range of IP Address Devices".
PCM_OP_IP_DEVICE_DELETE	Recommended. Deletes one or more IP address devices. See "Deleting an IP Address Device".
PCM_OP_IP_DEVICE_SET_ATTR	Recommended. Sets the attributes for one or more IP address devices. See "Modifying an IP Address Device".
PCM_OP_IP_DEVICE_SET_STATE	Recommended. Sets the device state for one or more IP devices. See "Changing IP Device States from Unallocated to Returned".

Job FM Policy Opcodes

Use the Job policy opcodes to validate and perform post-processing on business operations job definitions.

Include the **ops/job.h** header file in all applications that call these opcodes.

Table 1-41 Job FM Policy Opcodes

Opcode	Description
PCM_OP_JOB_POL_VALIDATE_DEFINITION	Recommended. Validates the contents of a job definition. See "Validating Job Definitions".
PCM_OP_JOB_POL_POST_PROCESS_DEFINITION	Recommended. Performs post-processing validation and actions on job definitions and related templates. See "Post-Processing Job Definitions".

Job FM Standard Opcodes

Use the Job standard opcodes to run BRM applications and business operations jobs.

Include the **ops/job.h** header file in all applications that call these opcodes.

Table 1-42 Job FM Standard Opcodes

Opcode	Description
PCM_OP_JOB_EXECUTE	Recommended Runs business operations jobs or BRM applications. See the following: • "How Business Operations Center Runs Jobs" in *BRM System Administrator's Guide*. • "Running Business Operation Jobs" • "Running BRM Applications"
PCM_OP_JOB_PROCESS_TEMPLATE	Recommended Creates, modifies, or deletes **/job_template** objects. See "Managing Business Operations Job Templates".
PCM_OP_JOB_PROCESS_DEFINITION	Recommended Creates, modifies, or deletes **/job_definition** and **/job_template/custom** objects. See the following: • Creating Custom Jobs • Modifying Custom Jobs • Deleting Custom Jobs

LDAP Base Opcodes

The LDAP opcodes are the base opcodes as implemented by LDAP Manager. These Base opcodes may be used by any of the opcodes in the BRM system to perform basic operations. Unlike all other opcodes, which belong to the Connection Manager, the base opcodes are part of the Data Manager.

For more information, see *BRM LDAP Manager* and "Base Opcodes".

Include the **ops/base.h** header file in all applications that call these opcodes.

Table 1-43 LDAP Base Opcodes

Opcode	Description
PCM_OP_CREATE_OBJ	Recommended. Creates new directory server entries or reuses entries in the directory server for replication purposes.
PCM_OP_DELETE_FLDS	Recommended. Deletes values and attributes in a directory server entry. This opcode performs the delete operation by using the LDAP *modify* operation, which imposes delete semantics.
PCM_OP_DELETE_OBJ	Recommended. Deletes an entire entry from the directory server. Invokes the delete entry semantics of the LDAP modify operation in the directory server.
PCM_OP_READ_FLDS	Recommended. Reads attributes from a directory server entry from the database using the LDAP search operation.

Table 1-43 (Cont.) LDAP Base Opcodes

Opcode	Description
PCM_OP_READ_OBJ	Recommended. Reads an entire object from the database using the LDAP search operation.
PCM_OP_SEARCH	Recommended. Searches the directory server based on a specified search criteria. *Only* those objects and attributes that you define in the mapping file can be returned by the LDAP Data Manager in the output flist.
PCM_OP_TEST_LOOPBACK	Recommended. Tests directory server connections. Verifies that the LDAP Data Manager and the directory server daemon/service processes are running and communicating with each other.
PCM_OP_WRITE_FLDS	Recommended. Updates attributes in a directory server entry or renames entries. This opcodes performs the following operations: • Updates attributes by using the LDAP *modify* operation. • Uses the *replace* semantics of the LDAP *modify* operation. • Renames directory server entries.

Loan FM Policy Opcodes

Use the loan policy opcodes to customize loan eligibility, granting, recovery, and reset fields in loan profiles.

See "Configuring Loans" in *BRM Configuring and Collecting Payments*.

Include the **ops/loan.h** header file in all applications that call these opcodes.

Table 1-44 Loan FM Policy Opcodes

Opcode	Description
PCM_OP_LOAN_POL_ELIGIBILITY	Recommended. Enables you to customize the eligibility verification for loans. See "Customizing Loan Eligibility Checks."
PCM_OP_LOAN_POL_PRE_APPLY_LOAN	Recommended. Enables you to customize the loan grant process. See "Customizing Loan Grants."
PCM_OP_LOAN_POL_PRE_RECOVER_LOAN	Recommended. Enables you to customize the loan recovery process. See "Customizing Loan Recovery."
PCM_OP_LOAN_POL_RESET_CYCLE	Recommended Enables you to reset fields in the **/profile/loan** object. See "Resetting a Loan Cycle and Other Loan Profile Fields".

Loan FM Standard Opcodes

The loan standard opcodes verify loan eligibility, apply loans, retrieve loans, and manage the loan cycle.

See the "Configuring Loans" in *BRM Configuring and Collecting Payments*.

Include the **ops/loan.h** header file in all applications that call these opcodes.

Table 1-45 Loan FM Standard Opcodes

Opcode	Description
PCM_OP_LOAN_ELIGIBILITY	Recommended. Checks whether a customer is eligible for a requested loan. See "Checking a Customer's Eligibility for a Loan".
PCM_OP_LOAN_APPLY_LOAN	Recommended. Grants a loan to an eligible customer. See "Granting a Loan".
PCM_OP_LOAN_GET_LOAN	Recommended. Retrieves the list of loans for an account. See "Retrieving Loan Information".
PCM_OP_LOAN_POL_RESET_CYCLE	Recommended Resets the loan cycle date. See "Resetting a Loan Cycle and Other Loan Profile Fields".

Monitor FM Standard Opcodes

The monitor standard opcodes enable your customers to monitor the balance of a group of accounts in near real time.

See "Managing Balance Monitoring Groups" in *BRM Managing Customers*.

Include the **ops/monitor.h** header file in all applications that call these opcodes.

Table 1-46 Monitor FM Standard Opcodes

Opcode	Description
PCM_OP_MONITOR_ACCOUNT_HIERARCHY	Recommended. Adds members to hierarchy-type balance monitors when an account hierarchy changes. See the following: • Adding Members to Newly Created Sharing Groups Automatically • Adding Members to Hierarchy-Type Sharing Groups Automatically

Table 1-46 (Cont.) Monitor FM Standard Opcodes

Opcode	Description
PCM_OP_MONITOR_BILLING_HIERARCHY	Recommended. Adds members to paying responsibility-type monitors when changes occur in an account hierarchy. See the following: • Adding Members to Newly Created Sharing Groups Automatically • Adding Members to Paying Responsibility-Type Sharing Groups Automatically
PCM_OP_MONITOR_HIERARCHY_CLEANUP	Recommended. Removes members from hierarchy-type or paying responsibility-type monitors when an account hierarchy changes. See the following: • Adding Members to Newly Created Sharing Groups Automatically • Removing Members from Balance Monitor Groups
PCM_OP_MONITOR_PROCESS_BILLING_MONITORS	Recommended. Adds members to payment responsibility-type balance monitors. See the following: • Adding Members to Newly Created Sharing Groups Automatically • Adding Members to Payment Responsibility-Type Balance Monitors
PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS	Recommended. Adds members to hierarchy-type balance monitors. See the following: • Adding Members to Newly Created Sharing Groups Automatically • Adding Members to Hierarchy-Type Balance Monitors
PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS	Recommended. Adds the following members to the balance monitor: the parent subscription service and all member services. See "Adding Members to Newly Created Sharing Groups Automatically".
PCM_OP_MONITOR_SERVICE_HIERARCHY	Recommended. Adds members to subscription-type monitors when a subscription group changes. See "Updating Subscription-Type Monitors Automatically".
PCM_OP_MONITOR_SETUP_MEMBERS	Recommended. Automatically adds members to a balance monitor when it is first created. This opcode is a wrapper opcode that, according to monitor group type, calls other standard monitor opcodes. See "Adding Members to Newly Created Sharing Groups Automatically".
PCM_OP_MONITOR_UPDATE_MONITORS	Recommended. Generates these events: • **/event/billing/monitor/update** when an **/ordered_balgrp** object is created or modified. • **/event/billing/monitor/delete** when an **/ordered_balgrp** object is deleted. See "Adding a Monitor Group to a Member's /ordered_balgrp Object".

Notification FM Standard Opcodes

Use the notification standard opcodes to manage messages that are sent to your customers through external notification applications.

See "Sending Messages to Customers through External Notification Applications" in *BRM Managing Customers*.

Include the **ops/notification.h** header file in all applications that call these opcodes.

Table 1-47 Notification FM Standard Opcodes

Opcode	Description
PCM_OP_NOTIFICATION_CALC_DELIVERY_TIME	Recommended. Calculates the customer's message delivery time. See "Calculating the Notification Delivery Time".
PCM_OP_NOTIFICATION_CREATE_SPECIFICATION	Recommended. Creates a notification specification. See "Creating Notification Specifications".
PCM_OP_NOTIFICATION_GET_LASTNOTIFY_TSTAMP	Recommended. Retrieves a customer's message delivery time range and offset value. See "Retrieving Last Notification Time and Offset".
PCM_OP_NOTIFICATION_GET_SPECIFICATION	Recommended. Retrieves the specified notification specification. See "Retrieving Notification Specifications".
PCM_OP_NOTIFICATION_MODIFY_SPECIFICATION	Recommended. Modifies the specified notification specification. See "Modifying Notification Specifications".
PCM_OP_NOTIFICATION_PROCESS_NOTIFICATION	Recommended. Manages in-advance and post-expiration notifications. See "Managing In-Advance and Post-Expiration Notifications".
PCM_OP_NOTIFICATION_VERIFY_PUBLISH_REQUIRED	Recommended. Verifies a customer's opt-in or opt-out preferences for receiving notification messages. See "Verifying Customer Opt-In Preferences".

Notification FM Policy Opcodes

Use the notification policy opcodes to customize messages that are sent to your customers through external notification applications.

See "Sending Messages to Customers through External Notification Applications" in *BRM Managing Customers*.

Include the **ops/notification.h** header file in all applications that call these opcodes.

Table 1-48 Notification FM Standard Opcodes

Opcode	Description
PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION	Recommended. Enables you to customize the notification specification retrieval process. See "Customizing Notification Retrieval".
PCM_OP_NOTIFICATION_POL_PREP_SPECIFICATION	Recommended. Enables you to customize notification specifications. See "Preparing Notification Specifications".
PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION	Recommended. Enables you to customize validation of the notification specifications. See "Validating Notification Specifications".

Number Manager FM Policy Opcodes

Use the Number Manager policy opcodes to customize Number Manager.

See the discussion on Number Manager in *BRM Telco Integration*.

Include the **ops/num.h** header file in all applications that call these opcodes.

Table 1-49 Number Manager FM Policy Opcodes

Opcode	Description
PCM_OP_NUM_POL_CANONICALIZE	Recommended. Enables you to customize number normalization when receiving numbers from applications and outputting numbers to other opcodes or applications. See "Customizing Number Normalization".
PCM_OP_NUM_POL_DEVICE_ASSOCIATE	Recommended. Enables you to customize the rules for associating or disassociate a number and a service. See "Customizing How Numbers Are Associated with Services".
PCM_OP_NUM_POL_DEVICE_CREATE	Recommended. Enables you to customize number validation to make sure the number is unique in the database. See "Customizing Telephone Number Attributes".
PCM_OP_NUM_POL_DEVICE_DELETE	Recommended. Enables you to customize number deletion. See "Deleting a Number".
PCM_OP_NUM_POL_DEVICE_SET_ATTR	Recommended. Enables you to customize which digits in a number can be changed. See "Customizing How a Number Can Be Changed".

Number Manager FM Standard Opcodes

The Number Manager standard opcodes create and modify blocks of numbers, manage number quarantine, and manage number portability.

See the discussion on Number Manager in *BRM Telco Integration*.

Include the **ops/num.h** header file in all applications that call these opcodes.

Table 1-50 Number Manager FM Standard Opcodes

Opcode	Description
PCM_OP_NUM_CREATE_BLOCK	Recommended. Creates a block of telephone numbers. See "Creating Blocks of Numbers".
PCM_OP_NUM_MODIFY_BLOCK	Recommended. Modifies a block of numbers, for example, changes the block name or numbers. See "Modifying Blocks of Numbers".
PCM_OP_NUM_PORT_IN	Limited Creates a number device by using the provided number. See "Managing Number Portability".
PCM_OP_NUM_PORT_OUT	Limited Sets the status of the provided number device to **quarantine_port_out**. See "Managing Number Portability".
PCM_OP_NUM_QUARANTINE	Recommended. Either creates or deletes a **/schedule/device** object to manage the telephone number quarantine. See "Managing Number Quarantine".
PCM_OP_NUM_SPLIT_BLOCK	Recommended. Splits an existing block of numbers into two or more blocks. See "Splitting Blocks of Numbers".

Offer Profile FM Standard Opcode

The offer profile standard opcodes create and manage the offer profile.

See the discussion on managing offer profiles with offer profile opcodes in *BRM Setting Up Pricing and Rating*.

Include the **ops/offer_profile.h** header file in all applications that call these opcodes.

Table 1-51 Offer Profile FM Standard Opcode

Opcode	Description
PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE	Limited Creates, modifies, or deletes the offer profiles (**/offer_profile** objects).

Order FM Policy Opcodes

Use the order policy opcodes to customize order management.

See the discussion on managing orders in *BRM Developer's Guide*.

Include the **ops/device.h** header file in all applications that call these opcodes.

Table 1-52 Order FM Policy Opcodes

Opcode	Description
PCM_OP_ORDER_POL_ASSOCIATE	Recommended. Enables you to customize validation during order association and disassociation.
PCM_OP_ORDER_POL_CREATE	Recommended. Enables you to customize validation during order creation.
PCM_OP_ORDER_POL_DELETE	Recommended. Enables you to customize validation during order deletion.
PCM_OP_ORDER_POL_PROCESS	Recommended. Enables you to customize processing of the order response. This opcode is called by PCM_OP_ORDER_PROCESS.
PCM_OP_ORDER_POL_SET_ATTR	Recommended. Enables you to customize how to validate order attribute changes. This opcode is called by PCM_OP_ORDER_SET_ATTR.
PCM_OP_ORDER_POL_SET_STATE	Recommended. Enables you to customize order state changes. This opcode is called by PCM_OP_ORDER_SET_STATE. Policy opcodes called during state changes are specified in the **/config/order_state** object.

Order FM Standard Opcodes

The order standard opcodes create, delete, and update **/order** objects.

See the discussion on orders in *BRM Developer's Guide*.

Include the **ops/order.h** header file in all applications that call these opcodes.

Table 1-53 Order FM Standard Opcodes

Opcode	Description
PCM_OP_ORDER_ASSOCIATE	Recommended. Associates or disassociates an order with a master **/order** object.
PCM_OP_ORDER_CREATE	Recommended. Creates an **/order** object.
PCM_OP_ORDER_DELETE	Recommended. Deletes an **/order** object.
PCM_OP_ORDER_PROCESS	Recommended. Processes the response of the order.

Table 1-53 (Cont.) Order FM Standard Opcodes

Opcode	Description
PCM_OP_ORDER_SET_ATTR	Limited Sets attribute values for an **/order** object.
PCM_OP_ORDER_SET_STATE	Limited Sets the state for an **/order** object.
PCM_OP_ORDER_UPDATE	Recommended. Updates the state or attributes for an **/order** object.

Payment FM Policy Opcodes

Use the payment policy opcodes to manipulate A/R functions and collect payments from customers.

See *BRM Configuring and Collecting Payments* for more information.

Include the **ops/pymt.h** header file in all applications that call these opcodes.

Table 1-54 Payment FM Policy Opcodes

Opcode	Description
PCM_OP_PYMT_POL_APPLY_FEE	Recommended. Enables you to customize payment fees by preprocessing, filtering, and extending the information available in failed payment fee events. See "Applying Payment Fees" and "Customizing Payment Fees" for more information.
PCM_OP_PYMT_POL_CHARGE	Limited. Enables you to map the online and offline payment result to the payment status and the reason IDs defined in the **/strings** object. See "BRM-Initiated Payment Processing" for more information.
PCM_OP_PYMT_POL_COLLECT	Recommended. Enables you to customize the results of a credit card transaction. See "Processing the Results of Credit Card Transactions" for more information.
PCM_OP_PYMT_POL_EXEC_COLLECTIONS_ACTION	Recommended. Completes a promise-to-pay installment as part of the payment process. See "Completing Promise-to-Pay Installments during Payment Processing".
PCM_OP_PYMT_POL_GRANT_INCENTIVE	Recommended. Enables you to enrich the PCM_OP_PYMT_GRANT_INCENTIVE input flist by specifying additional event attributes used to determine whether a payment incentive is granted. See "Customizing Payment Incentives".

Table 1-54 (Cont.) Payment FM Policy Opcodes

Opcode	Description
PCM_OP_PYMT_POL_MBI_DISTRIBUTE	Recommended. Enables you to customize the default payment distribution logic to distribute the submitted account payment to multiple bill units. See "Allocating Externally Initiated Payments by Due Amount" for more information.
PCM_OP_PYMT_POL_OVER_PAYMENT	Recommended. Enables you to customize how to allocate payments when there is an overpayment. See "How BRM Selects the Items to Which Payments Are Applied" for more information.
PCM_OP_PYMT_POL_PRE_COLLECT	Recommended. Enables you to check a batch of charges and refunds for any amounts below minimums before charging and refunding customers. See "BRM-Initiated Payment Processing" and "Externally Initiated Payment Processing" for more information.
PCM_OP_PYMT_POL_PROVISION_INCENTIVE	Recommended. Enables you to customize how to determine the payment date that should be considered when provisioning incentives. See "Customizing Payment Incentives".
PCM_OP_PYMT_POL_PURCHASE_DEAL	Recommended. Enables you to customize how to apply custom discounts and incentives to account balances when an account is topped up. See "Processing Top-Ups" for more information.
PCM_OP_PYMT_POL_SPEC_COLLECT	Recommended. Enables you to customize how much should be collected from an account after a specified action has been performed. See "Processing Credit Card Information during Account Creation" for more information.
PCM_OP_PYMT_POL_SPEC_VALIDATE	Recommended. Enables you to customize how to change the account used for credit card validation. See "Validating Credit Card and Direct Debit Transactions" for more information.
PCM_OP_PYMT_POL_SUSPEND_PAYMENT	Recommended. Enables you to customize how to send a payment marked for suspense to the payment suspense account. See "Validating Payments" for more information.
PCM_OP_PYMT_POL_UNDER_PAYMENT	Recommended. Enables you to customize how to allocate payments when there is an underpayment. See "How BRM Selects the Items to Which Payments Are Applied" for more information.

Table 1-54 (Cont.) Payment FM Policy Opcodes

Opcode	Description
PCM_OP_PYMT_POL_VALID_VOUCHER	Recommended. Enables you to customize how to interact with voucher management systems to validate vouchers. See "How BRM Handles Manual Standard Top-Ups" and the discussion on vouchers in *BRM Telco Integration*.
PCM_OP_PYMT_POL_VALIDATE	Recommended. Enables you to customize the result of validating a credit card transaction, including a description of that result. See "Validating Credit Card and Direct Debit Transactions" for more information.
PCM_OP_PYMT_POL_VALIDATE_PAYMENT	Recommended. Enables you to customize how to validate payments to determine whether they can be successfully posted or whether a failed, unconfirmed payment needs reversal. See "Validating Payments" for more information.

Payment FM Standard Opcodes

The payment standard opcodes collect payments and validate payment methods.

See *BRM Configuring and Collecting Payments* for more information.

Include the **ops/pymt.h** header file in all applications that call these opcodes.

Table 1-55 Payment FM Standard Opcodes

Opcode	Description
PCM_OP_PYMT_APPLY_FEE	Recommended. Creates payment fees for payments that fail, for example, due to insufficient account funds or an expired credit card. See "Processing Payment Fees" for more information.
PCM_OP_PYMT_CHARGE	Limited. Performs a BRM-initiated payment transaction. See: • BRM-Initiated Payment Processing • Initiating a Charge • Processing Credit Card Charges • Processing Direct Debit Charges
PCM_OP_PYMT_CHARGE_CC	Last Resort. Performs an online credit card transaction. See "Processing Credit Card Charges" for more information.
PCM_OP_PYMT_CHARGE_DD	Recommended. Performs a batch of Paymentech direct debit transactions. See "Processing Direct Debit Charges" for more information.

Table 1-55 (Cont.) Payment FM Standard Opcodes

Opcode	Description
PCM_OP_PYMT_CHARGE_DDEBIT	Recommended. Performs a debit card transaction. This opcode is used for the Paymentech direct debit implementation. See "About Paymentech Direct Debit Implementation".
PCM_OP_PYMT_COLLECT	Recommended. Performs payment collections and refunds. See: • Collecting Payments • Allocating Account Payments to Multiple Bill Units
PCM_OP_PYMT_FIND_TOPUP_EVENTS	Limited. Finds the **/event/billing/adjustment/account** event associated with sponsored top-ups. See "Finding Top-Up Events".
PCM_OP_PYMT_GET_ACH_INFO	Recommended. Retrieves the database ID of the DM interfacing with the automated clearing house using available information such as vendor name or element ID in the **/config/ach** object. See "Processing Credit Card Information during Account Creation" for more information.
PCM_OP_PYMT_GRANT_INCENTIVE	Limited. Applies a payment incentive to a bill during the billing run. See "Granting Payment Incentives" for more information.
PCM_OP_PYMT_ITEM_SEARCH	Limited. Searches for **/item** objects with a variable number of input parameters. This opcode calls PCM_OP_SEARCH based on the input argument fields. See "Finding Payments" for more information.
PCM_OP_PYMT_MBI_DISTRIBUTE	Limited. Distributes the account payment to multiple bill units. See "Allocating Externally Initiated Payments by Due Amount" for more information.
PCM_OP_PYMT_MBI_ITEM_SEARCH	Limited. Gets all the items of the bill units in a tree view. See "Allocating Account Payments to Multiple Bill Units".
PCM_OP_PYMT_PROVISION_INCENTIVE	Limited. Evaluates a payment to determine whether a payment incentive should be provisioned and, if so, sets the payment incentive trigger. See "Processing Payment Incentives" for more information.
PCM_OP_PYMT_RECOVER	Recommended. Checks results of charges sent in a batch and posts results of charges for which no information was returned. See "Checking the Results of BRM-Initiated Batch Payment Operations" for more information.

Table 1-55 (Cont.) Payment FM Standard Opcodes

Opcode	Description
PCM_OP_PYMT_RECOVER_CC	Limited. Checks results of credit card charges sent in a batch and posts results of credit card charges for which no information was returned. See "Checking the Results of BRM-Initiated Batch Payment Operations" for more information.
PCM_OP_PYMT_RECOVER_DD	Limited. Checks results of direct debit charges sent in a batch. The results are passed back and used for transaction reconciliation. This opcode is specific to the Paymentech DM. See "Checking the Results of BRM-Initiated Batch Payment Operations" for more information.
PCM_OP_PYMT_RECYCLE_PAYMENT	Limited. Processes payment reversals during payment recycling and assigns action owner codes to suspended payments. See "How Payments Are Recycled to and from Suspense" for more information.
PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH	Limited. Searches for recycled payments that have not been reversed, including those recycled to the payment suspense account. See "Retrieving Recycled Payments" for more information.
PCM_OP_PYMT_REVERSE_INCENTIVE	Limited. Reverses a payment incentive, provided the incentive has not yet been applied. See "Reversing Payment Incentives" for more information.
PCM_OP_PYMT_SELECT_ITEMS	Limited. Identifies a list of items based on the input fields and the accounting type of the account. See "How BRM Selects the Items to Which Payments Are Applied" for more information.
PCM_OP_PYMT_TOPUP	Recommended. Performs standard top-ups and sponsored top-ups. See "Processing Top-Ups" for more information.
PCM_OP_PYMT_VALIDATE	Recommended. Validates a credit card or direct debit transaction. See "Validating Payments" and "About the Default Payment Validation Process " for more information.
PCM_OP_PYMT_VALIDATE_CC	Limited. Performs a batch of credit card validations and applies the validation policy to the results. See "Validating Credit Card and Direct Debit Transactions" for more information.
PCM_OP_PYMT_VALIDATE_DD	Limited. Performs a batch of online direct debit validations and applies the validation policy to the results. See "Validating Credit Card and Direct Debit Transactions" for more information.

Table 1-55 (Cont.) Payment FM Standard Opcodes

Opcode	Description
PCM_OP_PYMT_VALIDATE_PAYMENT	Recommended. Validates payment records. See "Validating Payments" and "About the Default Payment Validation Process " for more information.

Permissioning FM Standard Opcodes

The permissioning standard opcodes create and manage Access Control Lists (ACLs), which specify the CSRs that can access customer accounts in an account group.

Include the **ops/perm.h** header file in all applications that call these opcodes.

Table 1-56 Permissioning FM Standard Opcodes

Opcode	Description
PCM_OP_PERM_ACL_GET_SUBGROUPS	Limited Retrieves a particular account group (**/group/billing** hierarchy) beneath the provided access control list. See "Managing ACL Groups".
PCM_OP_PERM_ACL_GROUP_CREATE	Limited Creates a **/group/acl** object.
PCM_OP_PERM_ACL_GROUP_DELETE	Limited Deletes a **/group/acl** object. Deleting an ACL does not affect services. It simply removes the ACL.
PCM_OP_PERM_ACL_GROUP_MODIFY	Limited Modifies a **/group/acl** object.
PCM_OP_PERM_ACL_GROUP_ADD_MEMBER	Limited Adds a group member to a **/group/acl**.
PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER	Limited Deletes a group member from a **/group/acl**.
PCM_OP_PERM_FIND	Limited Retrieves a list of Access Control Lists (ACLs) to which a CSR belongs and returns user specified information about each ACL. See "Finding CSR Membership".
PCM_OP_PERM_GET_CREDENTIALS	Limited Retrieves a list of ACLs and billing groups that can access an application.
PCM_OP_PERM_SET_CREDENTIALS	Limited Sets the current connection scope. Scope is defined by the billing group.

Process Audit FM Policy Opcodes

Use the process audit policy opcodes to create audit objects with revenue assurance data.

For more information about collecting revenue assurance data, see the discussion on understanding Revenue Assurance Manager in *BRM Collecting Revenue Assurance Data*.

Include the *BRM_home*/**include/ops/process_audit.h** header file in all applications that call these opcodes.

Table 1-57 Process Audit FM Policy Opcodes

Opcode	Description
PCM_OP_PROCESS_AUDIT_POL_CREATE	Recommended. Checks for duplicate audit objects and validates audit data. You can customize this opcode by modifying the fields in the flist, modifying duplicate checks, and adding validation checks. This opcode is called by PCM_OP_PROCESS_AUDIT_CREATE. The transaction handling for this opcode is done in the standard opcode.
PCM_OP_PROCESS_AUDIT_POL_ALERT	Recommended. Sends email messages when threshold values are crossed. You can customize this opcode to notify an external system and change the message body or subject of the email. This opcode is not called by any opcode.
PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY	Recommended. Called by PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY opcode to map the fields of **/suspended_usage/telco** to fields of **/process_audit/batchstat/status**. You can customize this opcode to read and aggregate any fields of the **/suspended_usage/***xxx* storable class, where *xxx* are subclasses of **/suspended_usage**, and map them to corresponding fields in the **/process_audit/batchstat/status** storable class. This opcode is called by PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY. The transaction handling for this opcode is done in the standard opcode.

Process Audit FM Standard Opcodes

The process audit standard opcodes are called by billing utilities to create audit objects with revenue assurance data.

For more information about collecting revenue assurance data, see the discussion on understanding Revenue Assurance Manager in *BRM Collecting Revenue Assurance Data*.

Include the *BRM_home*/**include/ops/process_audit.h** header file in all applications that call these opcodes.

Table 1-58 Process Audit FM Standard Opcodes

Opcode	Description
PCM_OP_PROCESS_AUDIT_CREATE	Recommended. Creates audit objects for revenue assurance. Called by the BRM billing applications.
PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY	Recommended. Creates **/process_audit/batchstat/status** objects with revenue assurance data for written-off records. Checks the **PIN_FLD_FLG** in the input flist during bulk suspense operations: • If **PIN_FLD_FLG** is zero, generates a normal summary. • If **PIN_FLD_FLG** is nonzero, creates a **/schedule** object after checking the mandatory fields. • If **PIN_FLD_FLG** is nonzero and the calling opcode is PCM_OP_ACT_SCHEDULE_EXECUTE, creates a bulk write-off summary.
PCM_OP_PROCESS_AUDIT_SEARCH	Recommended. Retrieves summary and detail data for control points for **/process_audit/batchstat** objects and subclasses.

Provisioning FM Policy Opcode

Use the provisioning policy opcode to customize provisioning.

See the discussion on managing GSM service provisioning in *BRM Managing Customers*.

Include the **ops/prov.h** header file in all applications that call these opcodes.

Table 1-59 Provisioning FM Policy Opcode

Opcode	Description
PCM_OP_PROV_POL_UPDATE_SVC_ORDER	Recommended. Enables you to customize how to validate and modify parameters for updating service orders. See "Managing GSM Service Provisioning".

Provisioning FM Standard Opcodes

The provisioning standard opcodes manage service order provisioning.

See the discussion on managing GSM service provisioning in *BRM Managing Customers*.

Include the **ops/prov.h** header file in all applications that call these opcodes.

Table 1-60 Provisioning FM Standard Opcodes

Opcode	Description
PCM_OP_PROV_PUBLISH_SVC_ORDER	Recommended. Sends a **/event/provisioning/service_order/*** event to the Provisioning Data Manager. See "Managing GSM Service Provisioning".
PCM_OP_PROV_UPDATE_SVC_ORDER	Recommended. Updates the status of an **/event/provisioning/service_order/*** event. See "Managing GSM Service Provisioning".

Publish FM Policy Opcodes

Use the publish policy opcodes to customize the notifications being published to the EAI Framework.

Include the **ops/fm_publish_pol.h** header file in all applications that call these opcodes.

Table 1-61 Publish FM Policy Opcode

Opcode	Description
PCM_OP_PUBLISH_POL_PREP_EVENT	Recommended. Enables you to customize the notifications that are published to the EAI Framework. See "Customizing Notification Enrichment" in *BRM Developer's Guide*.

Rating FM Policy Opcodes

The rating policy opcodes are called by Activity FM opcodes to calculate charges and taxes for an event.

Include the **ops/rate.h** header file in all applications that call these opcodes.

Table 1-62 Rating FM Policy Opcodes

Opcode	Description
PCM_OP_RATE_POL_GET_TAXCODE	Recommended. Enables you to customize the tax code information from the **taxcodes.map** file. See "Retrieving a List of Tax Codes".
PCM_OP_RATE_POL_GET_TAX_SUPPLIER	Recommended. Enables you to customize a list of tax suppliers retrieved from the **/profile/tax_supplier** object. See "Retrieving a List of Tax Suppliers".
PCM_OP_RATE_POL_MAP_TAX_SUPPLIER	Recommended. Enables you to customize how a tax supplier is chosen for an event. See "Retrieving a List of Tax Suppliers".

Table 1-62 (Cont.) Rating FM Policy Opcodes

Opcode	Description
PCM_OP_RATE_POL_POST_RATING	Recommended. Enables you to modify rated **/event** objects, for example, change the G/L ID of an event. See "Modifying Rated Events".
PCM_OP_RATE_POL_PRE_RATING	Recommended. Enables you to modify subscription events before rating. See "Modifying Rated Events".
PCM_OP_RATE_POL_PROCESS_ERAS	Recommended. Enables you to add extended rating attribute (ERA) information to an event. See "Modifying ERAs".
PCM_OP_RATE_POL_POST_TAX	Recommended. Enables you to modify data after taxes are calculated. See "Modifying Tax Data After Calculating Taxes".
PCM_OP_RATE_POL_PRE_TAX	Recommended. Enables you to modify data before you send the data to the taxation DM for calculating taxes. See "Modifying Tax Data Before Calculating Taxes".
PCM_OP_RATE_POL_TAX_LOC	Last Resort Enables you to retrieve locations for an event, which are then used to establish jurisdictions for tax calculation. See "Retrieving Tax Location Data".

Rating FM Standard Opcodes

The rating standard opcodes are called by Activity FM opcodes to calculate charges and taxes for an event.

Include the **ops/rate.h** header file in all applications that call these opcodes.

Table 1-63 Rating FM Standard Opcodes

Opcode	Description
PCM_OP_RATE_EVENT	Last Resort Applies pricing and taxes to a subscription event. See "Subscription Rating Opcodes".
PCM_OP_RATE_GET_ERAS	Last Resort Retrieves the extended rating attribute (ERAs) for an event. See "Getting ERAs for an Event".
PCM_OP_RATE_GET_PRODLIST	Last Resort Retrieves the list of charge offers owned by an account and filters them with input criteria. See "Subscription Rating Opcodes".

Table 1-63 (Cont.) Rating FM Standard Opcodes

Opcode	Description
PCM_OP_RATE_TAX_CALC	Last Resort Calculates taxes due at the time of purchase or billing. See "How PCM_OP_RATE_TAX_CALC Calculates Taxes".
PCM_OP_RATE_TAX_EVENT	Last Resort Directs the calculation of taxes. See "Subscription Rating Opcodes".

Remittance FM Policy Opcodes

Use the remittance policy opcode to customize remittance criteria.

Include the **ops/remit.h** header file in all applications that call these opcodes.

Table 1-64 Remittance FM Policy Opcodes

Opcode	Description
PCM_OP_REMIT_POL_SPEC_QTY	Recommended. Retrieves the quantity to be rated for a customized RUM. See "Customizing Remittance" in *BRM Configuring and Running Billing*.

Remittance FM Standard Opcodes

The remittance standard opcodes manage remittance.

Include the **ops/remit.h** header file in all applications that call these opcodes.

Table 1-65 Remittance FM Standard Opcodes

Opcode	Description
PCM_OP_REMIT_GET_PROVIDER	Limited Retrieves a list of remittance accounts that need to be remitted when a particular event occurs. See "Retrieving Remittance Accounts" in *BRM Configuring and Running Billing*.
PCM_OP_REMIT_REMIT_PROVIDER	Limited Calculates the remittance amount. See "Calculating the Remittance Amount" in *BRM Configuring and Running Billing*.
PCM_OP_REMIT_VALIDATE_SPEC_FLDS	Limited Validates remittance criteria. See "Verifying the Remittance Specification File" in *BRM Configuring and Running Billing*.

Replication FM Policy Opcode

The replication policy opcode implements the translation logic for **/account** and **/service** objects.

Include the **ops/repl.h** header file in all applications that call these opcodes.

Table 1-66 Replication FM Policy Opcode

Opcode	Description
PCM_OP_REPL_POL_PUSH	Recommended. Enables you to implement the translation logic for **/account** and **/service** objects. This opcode is the BRM interface for the LDAP Data Manager mapping operations. By default, the LDAP Data Manager implements a single-entry mapping operation. See the discussion on understanding the replication policy push operation in *BRM LDAP Manager*.

Rerating FM Standard Opcode

The rerating standard opcode calls other standard opcodes to create **/job/rerate** objects and **/job_batch/rerate** objects.

Include the **ops/rerate.h** header file in all applications that call these opcodes.

Table 1-67 Rerating FM Standard Opcode

Opcode	Description
PCM_OP_RERATE_INSERT_RERATE_REQUEST	Recommended. Calls other opcodes to create **/job/rerate** objects and **/job_batch/rerate** objects. See "How BRM Creates Rerate Jobs".

SDK FM Standard Opcodes

The SDK standard opcodes add, delete, and modify data dictionary components, including opcode mapping, storable classes, and fields.

See the discussion on the BRM SDK in *BRM Developer's Guide*.

Include the **ops/sdk.h** header file in all applications that call these opcodes.

Table 1-68 SDK FM Standard Opcodes

Opcode	Description
PCM_OP_SDK_DEL_FLD_SPECS	Limited. Deletes field specifications from all database schemas in your BRM system. **Caution:** If you delete field specifications for fields that have been instantiated, you will corrupt your database. For example, never delete PIN_FLD_POID from a base BRM system. Because of this danger, we recommend that you do not use this opcode on a production system.
PCM_OP_SDK_DEL_OBJ_SPECS	Limited. Deletes storable class specifications from the data dictionary of all database schemas in your BRM system. **Caution:** If you delete a storable class that has been instantiated, you will corrupt your database. For example, never delete the ***/*account** object. Because of this danger, we recommend that you *do not* use this opcode on a production system.
PCM_OP_SDK_GET_FLD_SPECS	Limited. Retrieves one or more field specifications.
PCM_OP_SDK_GET_OBJ_SPECS	Limited. Retrieves one or more storable class specifications.
PCM_OP_SDK_SET_FLD_SPECS	Limited. Creates or modifies field specifications. **Caution:** If you change field specifications for fields that have been instantiated, you will corrupt your database. Instead of using this opcode, it is safer and more reliable to create or modify field specifications by using the Storable Class Editor in Developer Center.
PCM_OP_SDK_SET_OBJ_SPECS	Limited. Creates or modifies a storable class. **Caution:** If you change field specifications for fields that have been instantiated, you will corrupt your database. Instead of using this opcode, it is safer and more reliable to create or modify field specifications by using the Storable Class Editor in Developer Center.

Services Framework Manager FM Policy Opcodes

The services framework opcodes are used for provisioning any prepaid service type.

See the discussion on the BRM SDK in *BRM Developer's Guide*.

Include the **ops/tcf.h** header file in all applications that call these opcodes.

Table 1-69 Services Framework Manager FM Policy Opcodes

Opcode	Description
PCM_OP_TCF_POL_APPLY_PARAMETER	Recommended. Enables you to map service types to specific opcodes when associating APNs and QoS values with GPRS services. See "Mapping Service Types to Service-Specific Opcodes".

Table 1-69 (Cont.) Services Framework Manager FM Policy Opcodes

Opcode	Description
PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER	Recommended. Enables you to perform custom actions to a provisioning service order before it is passed to **dm_prov_telco**. See "Provisioning Process Opcode Flow".

Services Framework Manager FM Provisioning Opcodes

The service framework provisioning opcodes perform provisioning functions.

See the discussion on provisioning in *BRM Telco Integration*.

Include the **ops/tcf.h** header file in all applications that call these opcodes.

Table 1-70 Services Framework Manager FM Provisioning Opcodes

Opcode	Description
PCM_OP_TCF_APPLY_PARAMETER	Recommended. Updates the objects impacted by the charge offer provisioning update. See "Associating APNs and QoS with GPRS Services".
PCM_OP_TCF_SVC_LISTENER	Recommended. Checks the action's start and end date and performs one of the following: • If it is a current action, this opcode calls PCM_OP_TCF_APPLY_PARAMETER. • If the action is deferred for the future, this opcode creates a **/schedule** object for executing PCM_OP_TCF_APPLY_PARAMETER at the scheduled time. See "Associating APNs and QoS with GPRS Services".
PCM_OP_TCF_PROPAGATE_STATUS	Recommended. When the service status changes to inactive, active, or closed, this opcode propagates the status to any associated features and extended rating attributes (ERAs). When the status changes to closed, this opcode also disassociates any existing devices from the service.
PCM_OP_TCF_PROV_CREATE_SVC_ORDER	Recommended. Creates service orders for provisioning prepaid services, devices, and profiles. See "Provisioning Process Opcode Flow".
PCM_OP_TCF_PROV_HANDLE_SVC_ORDER	Recommended. Prepares provisioning event data for publishing to the provisioning DM and initiates status change to PROVISIONING. See "Provisioning Process Opcode Flow".

Table 1-70 (Cont.) Services Framework Manager FM Provisioning Opcodes

Opcode	Description
PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT	Recommended. Updates the status of the **/service/device** object upon receiving the response from the provisioning platform. See "Provisioning Process Opcode Flow".
PCM_OP_TCF_PROV_UPDATE_SVC_ORDER	Recommended. Updates the **/event/provisioning/service_order/telco** object with the provisioning response specified in the opcode input flist. See "Provisioning Process Opcode Flow".
PCM_OP_TCF_PROV_SERVICE_ORDER_NOTIFY	Recommended. Performs service order state changes. See "Provisioning Process Opcode Flow".
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_ATTR	Recommended. Updates a service order object. See "Provisioning Process Opcode Flow".
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE	Recommended. Sets the state for service order objects and validates the service order state transition using information stored in the configuration object. See "Provisioning Process Opcode Flow".

SIM Manager FM Policy Opcodes

Use the SIM Manager policy opcodes to customize SIM Manager.

See the discussion on SIM Card Manager in *BRM Telco Integration*.

Include the **ops/sim.h** header file in all applications that call these opcodes.

Table 1-71 SIM Manager FM Policy Opcodes

Opcode	Description
PCM_OP_SIM_POL_DEVICE_ASSOCIATE	Recommended. Enables you to customize how SIM cards and services are associated. See "Customizing SIM Card Number Changes".
PCM_OP_SIM_POL_DEVICE_CREATE	Recommended. Enables you to customize validation rules for creating SIM card devices. See "Customizing SIM Card Validation".
PCM_OP_SIM_POL_DEVICE_SET_ATTR	Recommended. Enables you to customize how SIM cards are associated with services. See "Customizing SIM Card Number Changes".

Table 1-71 (Cont.) SIM Manager FM Policy Opcodes

Opcode	Description
PCM_OP_SIM_POL_ORDER_CREATE	Recommended. Enables you to customize how to validate SIM card orders. See "Creating SIM Cards".

SIM Manager FM Standard Opcodes

The SIM Manager standard opcodes create and manage SIM card objects in the BRM database.

See the discussion on SIM Card Manager in *BRM Telco Integration*.

Include the **ops/sim.h** header file in all applications that call these opcodes.

Table 1-72 SIM Manager FM Standard Opcodes

Opcode	Description
PCM_OP_SIM_CREATE_ORDER	Recommended. Creates a SIM card order object (**/order/sim**). See "Creating and Updating SIM Card Orders".
PCM_OP_SIM_DEVICE_PROVISION	Recommended. Moves the device state from New to Provisioning, associates a service, and disassociates the pre-provisioning service. See "Provisioning SIM Cards".
PCM_OP_SIM_PROCESS_ORDER_RESPONSE	Recommended. Processes a vendor response file. See "Creating SIM Cards".
PCM_OP_SIM_UPDATE_ORDER	Recommended. Updates an existing SIM card order object (**/order/sim**). See "Creating and Updating SIM Card Orders".

Subscription Management FM Policy Opcodes

Use the subscription management policy opcodes to customize subscription services.

Include these header files in all applications that call these opcodes:

- **ops/subscription.h**
- **ops/bill.h**

See the header file for a list of the Subscription Management FM opcodes defined in that file.

Table 1-73 Subscription Management FM Policy Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS	Recommended. Enables you to customize how to add sharing groups to ordered balance groups. See "Adding Members to a Profile Sharing Group".
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE	Recommended. Enables you to customize how to add ordered balance groups. See "Adding Members to a Profile Sharing Group".
PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING	Limited. Enables you to clear fields in a **/service** object when you cancel a charge offer. See "Customizing Provisioning When Canceling a Charge Offer".
PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST	Recommended. Enables you to customize automatic rerating or handle your own automatic rerating scenarios. See "How BRM Creates Rerate Jobs".
PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS	Limited. Enables you to customize provisioning. See "Getting a List of Provisioning Tags".
PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS	Recommended. Enables you to filter or return a subset of **/sponsorship** objects from the list of all available **/sponsorship** objects. See "Getting a List of Charges Available for Charge Sharing".
PCM_OP_SUBSCRIPTION_POL_POST_PROMO_EVENT	Recommended. Enables you to customize how promotions are applied to accounts. See "Customizing Special Date, Event, or Action Promotions".
PCM_OP_SUBSCRIPTION_POL_PRE_FOLD	Limited. Enables you to customize how folds are applied to currency and noncurrency balances. See "Customizing How Folds Are Applied".
PCM_OP_SUBSCRIPTION_POL_PRE_PROMO_EVENT	Recommended. Enables you to customize how promotions are rated. See "Customizing Special Date, Event, or Action Promotions".
PCM_OP_SUBSCRIPTION_POL_PREP_FOLD	Recommended. Enables you to customize the list of balances that must be folded when a charge offer is canceled. See "Customizing Which Balances to Fold When Charge Offers Are Canceled".
PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS	Recommended. Enables you to customize how to validate the members of a profile sharing group. See "Validating Profile Sharing Group Members".

Table 1-73 (Cont.) Subscription Management FM Policy Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL	Recommended. Enables you to customize validation based on account data during bundle-to-bundle transition. See "Customizing Bundle Transitions".
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN	Recommended. Enables you to customize validation based on account data during package-to-package transition. See "Customizing Package Transitions without Configuring / transition Objects".
PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING	Limited. Enables you to customize provisioning when a charge offer is purchased. See "Customizing Provisioning When a Charge Offer Is Purchased".
PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT	Recommended. Enables you to specify the distribution of group discounts to group members. See "Customizing Snowball Discounts".
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL	Recommended. Enables you to customize how charge offers are canceled. See "Customizing Charge Offer Cancellation".
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT	Recommended. Enables you to customize how discount offers are canceled. See "Customizing Discount Offer Cancellation".
PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL	Recommended. Enables you to customize the time interval for applying cycle forward and cycle arrears fees for a specified charge offer. See "Customizing Accounting Cycles."
PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD	Recommended. Enables you to specify the order in which to fold balances in a balance group. See "Customizing the Order to Apply Folds."
PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE	Recommended. Enables you to define custom searches for rerating events for selected accounts. See "Customizing Event Searches for Rerating."
PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING	Recommended. Enables you to define custom validation of product specification attributes for pricing components. See "Validating Product Specification Attributes for Pricing Components."

Subscription Management FM Standard Opcodes

The subscription management standard opcodes manage subscription services.

Include these header files in all applications that call these opcodes:

- **ops/subscription.h**
- **ops/bill.h**
- **ops/cust.h**

See the header file for a list of the Subscription Management FM opcodes defined in that file.

Table 1-74 Subscription Management FM Standard Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_CANCEL_DEAL	Recommended. Cancels ownership of a bundle for a specified account or service. See "How Bundles Are Canceled".
PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT	Recommended. Cancels the discount instances associated with the **/account** object or **/service** object specified in the input flist. See "How Discount Offers Are Canceled".
PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT	Recommended. Cancels the charge offers for the **/account** object specified in the input flist. See "How Charge Offers Are Canceled".
PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION	Recommended. Cancels a subscription service and its member services.
PCM_OP_SUBSCRIPTION_CHANGE_DEAL	Recommended. Changes the charge offers associated with a bundle for an account. See "How Bundles Are Modified".
PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS	Recommended. Validates changes to bundles (for example, to check for prerequisite or mutually exclusive bundles). See "Validating Changes to Bundles".
PCM_OP_SUBSCRIPTION_CRUD_OFFER_OVERRIDE	Recommended. Specifies how and when to override charges and discounts during specified time periods. See "Overriding Charges and Discounts for a Period of Time".
PCM_OP_SUBSCRIPTION_CYCLE_ARREARS	Last Resort. Applies cycle arrears charges to an account. See "Applying Cycle Arrears Fees".
PCM_OP_SUBSCRIPTION_CYCLE_FOLD	Last Resort. Applies cycle fold events for an account. If successful, it returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/fold** event. See "Applying Folds".
PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Last Resort. Applies cycle forward charges or refunds to an account. See "Applying Cycle Forward Fees".

Table 1-74 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_GET_HISTORY	Recommended. Retrieves the event history for a bundle, charge offer, discount offer, or service instance associated with an account. See "Finding Events Associated with Bundles, Charge Offers, Discount Offers, and Services".
PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS	Recommended. Retrieves details of price tags. See "Getting Information about Price Tags".
PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS	Recommended. Retrieves the purchased charge offers and discount offers associated with an account. See "Reading Data for All Valid Purchased Charge Offers and Discount Offers".
PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT	Recommended. Applies promotions to accounts. See "Applying Promotions for Special Dates, Events, or Actions".
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP	Recommended. Creates, modifies, or deletes the ordered balance group (**/ordered_balgrp** object) for an account or service that is a member of a charge, discount, or profile sharing group. See "Managing Ordered Balance Groups".
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY	Recommended. Creates one or more ordered balance groups (**/ordered_balgrp** objects) for an account or service. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".
PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE	Recommended. Creates the **/rate_change** object, which stores details about the charge offers affected by a pricing change, including the price tiers and charges in the charge offer. See "Rerating Cycle Forward and Cycle Forward Arrears Events".
PCM_OP_SUBSCRIPTION_PROVISION_ERA	Recommended. This opcode creates, modifies, or deletes **/profile** objects. When specified in a **/config/provisioning_tag** object, this opcode runs when a charge offer or discount offer containing the provisioning tag is purchased or canceled. Profiles can store extended rating attributes (ERAs) and other custom attributes. See "Managing Provisioning" in *BRM Implementing Charging*.
PCM_OP_SUBSCRIPTION_PURCHASE_DEAL	Recommended. Purchases the charge offers and discount offers in a bundle for the account or service object specified in the input flist. See "How Bundles Are Purchased".

Table 1-74 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT	Limited Allows purchase of a discount offer in a bundle. **Important:** Do not call this opcode directly. This opcode is always called by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. See "How Discount Offers Are Purchased".
PCM_OP_SUBSCRIPTION_PURCHASE_FEES	Recommended. Applies deferred purchase fees for a charge offer with an expired purchase start time. See "Applying Deferred Charge Offer Purchase Fees".
PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT	Limited. Purchases a charge offer for an account or service. **Important:** Do not call this opcode directly. This opcode is always called by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. See "Purchasing Charge Offers".
PCM_OP_SUBSCRIPTION_RATE_CHANGE	Recommended. Creates rerating requests when there is a cycle forward or cycle forward arrears event pricing change in the middle of the current cycle. See "Rerating Cycle Forward and Cycle Forward Arrears Events".
PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS	Recommended. Retrieves the hierarchical relationships of bundles, charge offers, discount offers, and services for an account. See "Getting a List of Bundles, Charge Offers, Discount Offers, and Services".
PCM_OP_SUBSCRIPTION_RERATE_REBILL	Recommended. Rerates events for a specified account. See "Customizing Event Searches for Rerating".
PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER	Recommended. A wrapper opcode that performs all the tasks necessary to transfer a service from one balance group to another. See "Transferring Services between Balance Groups".
PCM_OP_SUBSCRIPTION_SET_BUNDLE	Recommended. Creates, modifies, and deletes **/purchased_bundle** objects. Use this opcode to add promotion names and details to BRM invoices. See "Managing Promotions with Siebel CRM".
PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO	Recommended. Modifies or sets a discount's purchase, cycle, or usage date information. See "How Discount Offer Purchase, Cycle, and Usage Validity Is Modified".

Table 1-74 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS	Recommended. Changes the status of a **/purchased_discount** object in a bundle for an account or service. See "How BRM Changes Discount Offer Status".
PCM_OP_SUBSCRIPTION_SET_PRODINFO	Recommended. Changes the information for a specified charge offer in an account. See "Changing the Purchase, Usage, and Cycle Start and End Times".
PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS	Recommended. Sets the charge offer status of a **/purchased_product** object owned by an account. See "How BRM Changes Charge Offer Status".
PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE	Recommended. Creates a discount sharing group, charge sharing group, or profile sharing group. See "Creating a Sharing Group".
PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE	Recommended. Deletes a discount sharing group or charge sharing group. See "Deleting a Sharing Group".
PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY	Recommended. Modifies sharing groups. See "Modifying a Sharing Group".
PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT	Recommended. Changes the owner of a discount sharing group, charge sharing group, or profile sharing group. See "Changing the Owner of a Sharing Group".
PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER	Recommended. Checks the rollover-transfer profile object, **/profile/rollover_transfer**, to make sure it is configured and valid for the balance and receiver, and then transfers the entire rollover amount to the receiver. See "Performing Rollover Transfers".
PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION	Recommended. Transfers a subscription service to another subscriber account. See "Transferring Service Groups between Accounts in Different Schemas".
PCM_OP_SUBSCRIPTION_TRANSITION_DEAL	Recommended. Transitions a bundle from one account to another. See "How Bundles Are Transitioned".
PCM_OP_SUBSCRIPTION_TRANSITION_PLAN	Recommended. Transitions one package to another. See "How BRM Transitions Accounts from Source Packages to Target Packages".

Table 1-74 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description
PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY	Recommended. Validates bundle-to-bundle dependency rules. See "How Bundle Dependencies Are Validated".
PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY	Recommended. Validates a discount to see whether it can be used with other discounts or packages. Mutually exclusive dependencies are configured in the **/dependency** storable class. See "Validating Discount Offer Dependencies".

Suspense Manager FM Standard Opcodes

The Suspense Manager standard opcodes manage suspended event records stored in the BRM database as **/suspended_usage** objects. The opcodes are available only if you have purchased Suspense Manager.

For information about suspense manager, see *BRM Implementing Charging*.

Include the **ops/suspense.h** header file in all applications that call these opcodes.

Table 1-75 Suspense Manager FM Standard Opcodes

Opcode	Description
PCM_OP_SUSPENSE_DEFERRED_DELETE	Recommended. Deletes event records in a written-off state or a succeeded state. See "Deleting Suspended Records in Bulk".
PCM_OP_SUSPENSE_DELETE_USAGE	Recommended. Deletes event records in a written-off state or a succeeded state. See "Deleting Suspended Records".
PCM_OP_SUSPENSE_EDIT_USAGE	Recommended. Changes the contents of event record fields for a suspended call record. Suspense Management Center calls this opcode to edit a suspended call record. See "Changing the Contents of Fields in Suspended Event Records".
PCM_OP_SUSPENSE_RECYCLE_USAGE	Recommended. Initiates event record recycling. See "Initiating Suspense Recycling".
PCM_OP_SUSPENSE_SEARCH_DELETE	Recommended. Deletes call records with a status of **succeeded** or **written off** that match criteria specified in the input flist. See "Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off".
PCM_OP_SUSPENSE_SEARCH_EDIT	Recommended. Changes fields in a large number of suspended records in one database operation. See "Editing Suspended Records in Bulk".

Table 1-75 (Cont.) Suspense Manager FM Standard Opcodes

Opcode	Description
PCM_OP_SUSPENSE_SEARCH_RECYCLE	Recommended. Searches for and queues suspended call records for recycling based on criteria specified in the input flist. See "Recycling Suspended Records".
PCM_OP_SUSPENSE_SEARCH_WRITE_OFF	Recommended. Writes off a large number of suspended records in one database operation. See "Writing Off Suspended Records in Bulk".
PCM_OP_SUSPENSE_UNDO_EDIT_USAGE	Recommended. Undoes edits to suspended call records used by Suspense Manager. See "Undoing Edits to Suspended Event Records".
PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE	Recommended. Writes off suspended event records. When a suspended event record is written off, they cannot be edited or recycled. See "Writing Off Suspended Records in Bulk".

Universal Message Store (UMS) FM Standard Opcodes

The UMS standard opcodes support Universal Message Store functionality.

See the discussion of BRM message services in *BRM Developer's Guide* for more information.

Include the **ops/ums.h** header file in all applications that call these opcodes.

Table 1-76 Universal Message Store FM Standard Opcodes

Opcode	Description
PCM_OP_UMS_GET_MESSAGE	Recommended. Retrieves **/message** objects. An application that consumes messages from the UMS framework uses this opcode to retrieve messages that match the scope specified in the input flist.
PCM_OP_UMS_GET_MESSAGE_TEMPLATE	Recommended. Retrieves message templates from **/strings** objects. Applications that produce messages for the UMS framework call this opcode to retrieve message templates.
PCM_OP_UMS_GET_MESSAGE_TEMPLATES	Recommended. Retrieves a list of message templates.
PCM_OP_UMS_SET_MESSAGE	Recommended. Creates a **/message** object from the message template, filling in placeholders with supplied data.

2

Context Management Opcodes

Learn about the Oracle Communications Billing and Revenue Management (BRM) context management opcodes.

Topics in this document:

- Context Management Opcodes
- PCM_CONNECT
- PCM_CONTEXT_CLOSE
- PCM_CONTEXT_OPEN
- PCM_OP
- PCM_OPREF

Context Management Opcodes

The context management opcodes manage the communication between a client application and the BRM database.

Include the **pcm.h** header file in all applications that call these opcodes. See the discussion on header files in *BRM Developer's Guide*.

Table 2-1 lists the context management opcodes for BRM.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 2-1 Context Management Opcodes

Opcode	Description
PCM_CONNECT	Opens a PCM context in a BRM application.
PCM_CONTEXT_CLOSE	Closes a PCM context.
PCM_CONTEXT_OPEN	Opens a PCM context.
PCM_OP	Runs a PCM opcode by passing a copy of the input flist.
PCM_OPREF	Runs a PCM opcode by passing a reference to the input flist.

PCM_CONNECT

PCM_CONNECT simplifies opening a PCM context in a BRM application program. Instead of having to manually create an input flist for PCM_CONTEXT_OPEN, you can put all information necessary for opening a context in the application's **pin.conf** file. PCM_CONNECT reads those entries from the application's **pin.conf** file, creates an input flist with that information, and then calls PCM_CONTEXT_OPEN to open the context.

PCM_CONNECT first looks in the application's **pin.conf** file for the **userid** and **login_type** entries. If **login_type** is **0** (no login and password required), PCM_CONNECT calls PCM_CONTEXT_OPEN with a NULL input flist. The POID from the **userid** entry is used to route the context open request to the desired database schema. See "PCM_CONTEXT_OPEN" for details on **userid** values.

If **login_type** is **1**, PCM_CONNECT also reads the **login_name** and **login_pw** entries. It then calls PCM_CONTEXT_OPEN with an input flist containing values for PIN_FLD_POID, PIN_FLD_TYPE, PIN_FLD_LOGIN, and PIN_FLD_PASSWD_CLEAR, which it got from the **userid**, **login_type**, **login_name**, and **login_pw pin.conf** entries, respectively.

> **Note:**
>
> In your application, when you open a context and connect to the BRM server, run all the opcodes before closing the context. Connections add a significant overhead to the system, which affects performance. Therefore, to improve performance, run all the opcodes in an open context instead of opening and closing contexts frequently. Use CM proxy for applications that cannot maintain an open context for a long time. See the discussion on using **cm_proxy** to allow unauthenticated logins in *BRM System Administrator's Guide*.

See "PCM_CONTEXT_OPEN" for a full description of opening contexts.

Syntax

```
#include "pcm.h"
void
PCM_CONNECT(
                pcm_context_t     **ctxp,
                int64              *db_no,
                pin_errbuf_t       *ebufp);
```

Parameters

pcm_ctxp
A pointer to an open PCM context, which is returned by a successful call.

db_no
If **0** is passed in by using **(int64) 0**, **0** is returned. Otherwise, the number of the database schema to which this context has been opened is returned. The database number comes from the **userid** entry in the calling application's **pin.conf** file.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller.

Return Values

See "PCM_CONTEXT_OPEN".

Error-Handling

See "PCM_CONTEXT_OPEN".

Examples

The **sample_app.c** file and the accompanying makefile illustrate how to use this opcode when setting up a generic BRM account and service. The files are located in *BRM_SDK_home***/source/samples/app/c**.

Here is an additional example of how to use PCM_CONNECT:

```
PIN_ERR_CLEAR_ERR(&ebuf);
PCM_CONNECT(&ctxp, &db_no, &ebuf);
if (PIN_ERR_IS_ERR(&ebuf)) {
...
}
```

PCM_CONTEXT_CLOSE

PCM_CONTEXT_CLOSE closes a context to the BRM system. The context should be closed after it is no longer needed by an application. This opcode breaks the connection to the BRM system and frees all memory associated with the context.

If an application exits, all open contexts are automatically closed by the BRM system.

See the discussion on the PCM API in *BRM Developer's Guide* for more information on contexts.

Syntax

```
#include "pcm.h"
void
PCM_CONTEXT_CLOSE(
                pcm_context_t     *pcm_ctxp,
                int32              how,
                pin_errbuf_t      *ebufp);
```

Parameters

pcm_ctxpp
A pointer to an open PCM context.

how
The *how* parameter tells how to close the connection. The normal choice is to completely close the connection by passing in a **(int32) 0**. However, if you fork a process, the process that does not continue making PCM calls (usually the child process) should at least close all open FDs. This can be done by passing PCM_CONTEXT_CLOSE_FD_ONLY as the value of *how*. This has the benefit of allowing the child process (in most cases) to close the FDs without closing the PCM connection in the parent that spawned it. If the child process wants to continue to make PCM calls, it should open another PCM connection.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller.

Return Values

PCM_CONTEXT_CLOSE returns nothing.

Error status is passed back to the caller using the error buffer.

Error-Handling

PCM_CONTEXT_CLOSE uses individual-style ebuf error handling. This means the application must explicitly test for an error condition recorded in the error buffer before making other calls to the BRM API. See the discussion on understanding API error handling and logging for details on error handling algorithms in *BRM Developer's Guide*.

Examples

The **sample_app.c** file and the accompanying makefile illustrate how to use this opcode when setting up a generic BRM account and service. The files are located in *BRM_SDK_home***/source/samples/app/c**.

PCM_CONTEXT_OPEN

PCM_CONTEXT_OPEN opens a context to the BRM database. All data in the BRM database is accessed using an open context. A dynamically allocated context structure is passed back and is used in subsequent PCM calls to identify the open context. The context structure is opaque to the application and is used only to identify the context for other calls.

> **Note:**
>
> - In your application, when you open a context and connect to the BRM server, run all the opcodes before closing the context. Connections add a significant overhead to the system, which affects performance. To improve performance, run all the opcodes in an open context instead of opening and closing contexts frequently. If a context is opened from within a CM, it must remain open during the entire client life cycle. Use CM proxy for applications that cannot maintain an open context for a long time. See the discussion on using **cm_proxy** to allow unauthenticated logins in *BRM System Administrator's Guide*.
> - If you have client applications running on the same server as the CM or DM, you still must use a **TCP/IP** connection for invoking opcodes. You also must establish a PCM connection to obtain a context for invoking opcodes.

A context can only have one outstanding operation at a time. Even if the asynchronous routines are used to launch an operation, another one cannot be started until the outstanding one is either completed or canceled.

If parallel operations are required (in the same or a different database schema), the application can open multiple contexts to the BRM database. There is no limit to the number of contexts an application can open.

When a context is no longer needed, it should be closed using PCM_CONTEXT_CLOSE. The open context can survive any errors (except losing the socket), so it can still be used even after one operation has failed.

A single context is normally opened by a client to access a single database schema. The client application is responsible for including a POID in its PCM library calls. Each POID contains a database number. The CM uses this schema number to route the client's request (the operation) to the proper DM.

A single context can support access to many database schemas simultaneously, but the client is responsible for passing the correct database schema IDs. Furthermore, the CM handling requests for the client must be configured to access multiple database schemas. That is, it must have the schema numbers and IP addresses. This information is passed to the CM by using the **dm_pointer** entries in the CM's **pin.conf** file.

Only one transaction can be open at a time, and object manipulation functions performed while a transaction is open must apply to the same database schema. If a transaction is opened and you must access another schema, open another context and access it through the new context. See the discussion on the PCM API document in *BRM Developer's Guide* for more information on contexts.

For PCM_CONTEXT_OPEN inside an FM, always use **(pin_flist_t*)NULL** for *in_flistp*.

The BRM base database does not support transactions across database systems.

By default, CMs require a user login and password when requesting an open context using PCM_CONTEXT_OPEN. However, you can remove this authentication requirement by configuring the CM with a **cm_login_module** of **cm_login_null.so**. The **cm_login_module** entry in the CM's **pin.conf** file is explained in the comments in that file. When the CM is configured to require a password and login, the input flist (*in_flistp*) for PCM_CONTEXT_OPEN must be constructed as explained below.

Syntax

```
#include "pcm.h"
void
PCM_CONTEXT_OPEN(
               pcm_context_t     **pcm_ctxpp,
               pin_flist_t        *in_flistp,
               pin_errbuf_t       *ebufp);
```

Parameters

pcm_ctxpp
A pointer to an open PCM context.

in_flistp
Two types of login are supported:

- **type** = **0** base level security: verify the specified service by **type** and **ID**.
- **type** = **1** login/password security: look up the specified service by login name and validate the password.

If *in_flistp* is **NULL**, **type 0** login is attempted. Otherwise, the input flist can specify either **type 0** or **type 1** login.

For **type 0** login, the following two fields are required:

- **PIN_FLD_POID**

  The portions of the POID that are used during login verification are database number, service, and ID. The specified service with the specified ID is looked up in the BRM database. If this service does not exist, the login is denied. The root account's **/service/pcm_client** service can be used for the service and its ID of **1** can be used for the ID.

Any valid service **type** and **ID** could be used instead of the root account's **/service/pcm_client** service.

- **PIN_FLD_TYPE**

  The login type is **0**.

For **type 1** login, the following four fields are required:

- **PIN_FLD_POID**

  The portions of the POID used during login verification are database number, service, and ID. In the case of **type 1** login, the database number and service type are significant. The ID is required because the POID requires one, but any value can be used (usually **1**). The BRM database is searched for a service object (matching the service type contained in PIN_FLD_POID) that has a login that matches the login value for the PIN_FLD_LOGIN field. If no service with the specified login exists, the login is denied. Otherwise, the password is checked.

  The root account's **/service/pcm_client** service can be used for the service type, and its ID of **1** can be used for the ID. You are free to create other **/service** objects that can be used for login verification.

- **PIN_FLD_TYPE**

  The login **type** is **1**.

- **PIN_FLD_LOGIN**

  A login name.

  > **Note:**
  >
  > The login cannot contain the characters **:** and @. The **/** character is allowed.

- **PIN_FLD_PASSWD_CLEAR**

  The cleartext password for login.

  > **Note:**
  >
  > The password cannot contain the characters **:** and @. The **/** character is allowed.

See **PCM_CONTEXT_OPEN.input**, the input flist specification, for more details on *in_flistp*.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller.

Return Values

PCM_CONTEXT_OPEN returns nothing.

Error status is passed back to the caller using the error buffer.

The context structure used to identify the open context is passed back using *pcm_ctxpp*. If an error occurred, **NULL** is passed back.

Error-Handling

PCM_CONTEXT_OPEN uses individual-style ebuf error handling. This means the application must explicitly test for an error condition recorded in the error buffer before making other calls to the BRM API. See the discussion on understanding API error handling and logging for details on error handling algorithms in *BRM Developer's Guide*.

The following codes may be returned in **ebufp->pin_err**:

- **PIN_ERR_BAD_ARG**

  Indicates one of the following conditions:

  - The **flags** parameter was not set properly.
  - The PCM **ctxpp** or **ebufp** structures are NULL.
  - The configuration information does not point to a valid Connection Manager.
  - Unable to open a socket to the Connection Manager.
  - Too many sessions are open.
- **PIN_ERR_NONE**

  Opcode successful; operation is complete.
- **PIN_ERR_NO_MEM**

  A memory allocation failed.
- **PIN_ERR_BAD_LOGIN_RESULT**

  The login failed.

PCM_OP

PCM_OP is a wrapper function for all PCM opcodes. This opcode performs a PCM opcode operation on an open context. The operation is done synchronously, so the calling process waits until the operation is complete and has the return flist immediately available for inspection.

All PCM opcode operations can be performed using this opcode. The specific fields required and allowed on the input and return flists depend on the operation being performed.

If a PCM base opcode operation is run using PCM_OP when no transaction is open on the context, the operation is implicitly wrapped in a transaction so all effects of the operation occur atomically. If a PCM FM opcode operation is run when no transaction is open on the context, it may or may not implicitly wrap all changes in a transaction. This is dependent on the FM operation being performed.

Syntax

```
#include "pcm.h"
void
PCM_OP(
                pcm_context_t        *pcm_ctxp,
                pin_opcode_t          opcode,
                pin_opcode_flags_t    op_flags,
                pin_flist_t          *in_flistp,
                pin_flist_t         **ret_flistpp,
                pin_errbuf_t         *ebufp);
```

Parameters

pcm_ctxp
A pointer to an open PCM context.

opcode
The opcode to call. See "Base Opcodes" for choices.

op_flags
The flags supported by the opcode being called. See the opcode descriptions for information on the flags they take. Most opcodes take no flags, which is input as **(pin_opcode_flags_t) 0**.

in_flistp
A pointer to the input flist.

ret_flistpp
A pointer to a pointer for passing back the return flist. All opcodes produce a return flist with at least the PIN_FLD_POID field on it. Other fields on the return flist depend on the opcode being called. The return flist is passed back even if an error occurred during the operation. It is the responsibility of the caller to destroy the return flist when it is no longer needed.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller.

Return Values

PCM_OP returns nothing.

Error status is passed back to the caller using the error buffer.

The return flist is passed back using *ret_flistpp*. A return flist is always passed back, even if an error occurs. It is the responsibility of the caller to destroy both the input and return flists.

The following codes may be returned:

PIN_ERR_NONE

Opcode successful; operation is complete.

other codes

Opcode failed; see the "Error-Handling" section.

Error-Handling

PCM_OP uses individual-style ebuf error handling. This means the application must explicitly test for an error condition recorded in the error buffer before making other calls to the BRM API. See the discussion on understanding API error handling and logging for details on error handling algorithms in *BRM Developer's Guide*.

The following error codes returned from PCM_OP indicate an error in the PCP transmission protocol:

- PIN_ERR_BAD_XDR
- PIN_ERR_STREAM_EOF
- PIN_ERR_STREAM_IO
- PIN_ERR_TRANS_LOST
- PIN_ERR_CM_ADDRESS_LOOKUP_FAILED

If you see one of these errors, close the context on which the error occurred and open a new one. The output flist is undefined, but the input flist is still valid.

Examples

The **sample_app.c** file and the accompanying makefile illustrate how to use this opcode when setting up a generic BRM account and service. The files are located in *BRM_SDK_home*/**source/samples/app/c**.

PCM_OPREF

You use PCM_OPREF to call FM opcodes in the same way as PCM_OP. The opcode syntax and input parameters are the same as PCM_OP. The only difference between them is that PCM_OPREF passes a reference to the input flist whereas PCM_OP passes a copy of the input flist to the called opcode.

PCM_OPREF should be used to call opcodes that won't modify the input flist.

When you have large input flists (for example, invoice flists), using PCM_OPREF is a more efficient than PCM_OP because it passes the flist by reference and does not make a copy of the input flist, which saves memory.

Syntax

```
#include "pcm.h"
void
PCM_OPREF(
                pcm_context_t        *pcm_ctxp,
                pin_opcode_t          opcode,
                pin_opcode_flags_t    op_flags,
                pin_flist_t          *in_flistp,
                pin_flist_t         **ret_flistpp,
                pin_errbuf_t         *ebufp);
```

Parameters

pcm_ctxp
A pointer to an open PCM context.

opcode
The opcode to be performed. See "Base Opcodes" for choices.

op_flags
The flags supported by the opcode being called. See the opcode descriptions for information on the flags they take. Most opcodes take no flags, which is input as **(pin_opcode_flags_t) 0**.

in_flistp
A pointer to the input flist.

ret_flistpp
A pointer to a pointer for passing back the return flist. All opcodes produce a return flist with at least the PIN_FLD_POID field on it. Other fields on the return flist depend on the opcode being called. The return flist is passed back even if an error occurred during the operation. It is the responsibility of the caller to destroy the return flist when it is no longer needed.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller.

Return Values

PCM_OPREF returns nothing.

The return flist is passed back using *ret_flistpp*. A return flist is always passed back, even if an error occurs. It is the responsibility of the caller to destroy both the input and return flists.

Error status is passed back to the caller using the error buffer.

The following codes may be returned:

- PIN_ERR_NONE

  Opcode successful; operation is complete.

- *other codes*

  Opcode failed; see the "Error-Handling" section.

Error-Handling

PCM_OPREF uses individual-style ebuf error handling. This means the application must explicitly test for an error condition recorded in the error buffer before making other calls to the BRM API. See the discussion on understanding API error handling and logging for details on error handling algorithms in *BRM Developer's Guide*.

The following error codes returned from PCM_OPREF indicate an error in the PCP transmission protocol:

- PIN_ERR_BAD_XDR
- PIN_ERR_STREAM_EOF
- PIN_ERR_STREAM_IO
- PIN_ERR_TRANS_LOST
- PIN_ERR_CM_ADDRESS_LOOKUP_FAILED

If you see one of these errors, close the context on which the error occurred and open a new one. The output flist is undefined, but the input flist is still valid.

Part II
Opcode Workflows

This part describes Oracle Communications Billing and Revenue Management (BRM) opcode workflows.

Part II contains the following chapters:

- About Opcode Workflows
- Accounts Receivable Opcode Workflows
- Activity Opcode Workflows
- Billing Opcode Workflows
- Charging Opcode Workflows
- Collections Opcode Workflows
- Customer Management Opcode Workflows
- Deposit Opcode Workflows
- Device Management Opcodes
- General Ledger Opcode Workflows
- Group Opcode Workflows
- Installment Opcode Workflows
- Invoice Opcode Workflows
- Job Opcode Workflows
- Loan Opcode Workflows
- Notification Opcode Workflows
- Number Manager and SIM Card Manager Opcode Workflows
- Role Opcode Workflows
- Payment Opcode Workflows
- Subscription Opcode Workflows
- Tax Calculation Opcode Workflows
- Using the IP Address Manager APIs

3
About Opcode Workflows

Learn about Oracle Communications Billing and Revenue Management opcode workflow concepts. For a list of opcodes with descriptions, see "Opcode Descriptions".

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

An opcode workflow describes the opcodes that perform BRM operations (for example, creating accounts, purchasing bundles, and running billing). Therefore, in this document, workflow descriptions are organized not according to opcodes, but instead, according to types of BRM operations, such as billing and customer management.

An opcode workflow can describe:

- The main opcode that performs the operation. For example, PCM_OP_CUST_COMMIT_CUSTOMER is the main opcode for creating accounts.
- Additional opcodes that are called during the opcode flow. For example, before creating an account, PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_POL_PRE_COMMIT to enable customization.
- The actions the opcode performs (for example, retrieve data, change data, or delete data).
- When an opcode is called (for example, by another opcode).
- Important fields in input and output flist that can be used.
- Customizations that can be performed by using the opcodes.
- Conditions that cause an opcode to fail.

Use opcode workflows for the following purposes:

- To find out which opcodes to use when customizing BRM.
- To learn how BRM works.
- To troubleshoot BRM by tracking the progress of operations. For example, you can determine which opcode is failing if you know what the sequence of opcode calls is.

4

Accounts Receivable Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) accounts receivable (A/R) opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Getting a List of A/R Items
- Reversing Refunds
- Transferring Amounts between Items
- Managing Currency and Non-Currency Balance Transfers between Accounts or Services
- Transferring Services between Balance Groups
- Performing Adjustments
- Performing Disputes and Settlements
- Writing Off Debts and Reversing Write-Offs
- Retrieving A/R Information
- Performing Rollover Transfers

Opcodes Described in This Chapter

Table 4-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 4-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_SPEC_GLID	Assigning G/L IDs for an Adjustment Assigning G/L IDs for a Dispute or Settlement Performing Write Offs
PCM_OP_ACT_USAGE	Tax Processing for Account Adjustments
PCM_OP_AR_ACCOUNT_ADJUSTMENT	Transferring Balances between Balance Groups Adjusting Accounts, Subscription Services, and Member Services

Table 4-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_AR_ACCOUNT_WRITEOFF	About Account Write-Offs About Initiating Write-Offs Performing Write Offs About Bill Unit Write-Offs
PCM_OP_AR_BILL_ADJUSTMENT	Adjusting Bills Adjusting Items Tax Processing for Bill and Item Level Adjustments
PCM_OP_AR_BILL_CREDIT_TRANSFER	Transferring from a Negative to a Positive Balance
PCM_OP_AR_BILL_DISPUTE	Disputing Bills Disputing Items
PCM_OP_AR_BILL_SETTLEMENT	Settling Disputed Bills
PCM_OP_AR_BILL_WRITEOFF	About Bill Write-Offs About Initiating Write-Offs Performing Write Offs Flags You Should Use for Write-Offs
PCM_OP_AR_BILLINFO_WRITEOFF	About Bill Unit Write-Offs About Initiating Write-Offs Performing Write Offs About Account Write-Offs
PCM_OP_AR_EVENT_ADJUSTMENT	Adjusting Events Transferring Balances between Items Tax Processing for Account Adjustments Tax Processing for Bill and Item Level Adjustments Tax Processing for Event Adjustments Including Reason Codes in the Adjustment
PCM_OP_AR_EVENT_DISPUTE	Disputing Events Transferring Balances between Items
PCM_OP_AR_EVENT_SETTLEMENT	Settling Disputed Events Transferring Balances between Items
PCM_OP_AR_GET_ACCT_ACTION_ITEMS	Getting a List of A/R Items Retrieving A/R Items That Apply to a Bill Unit Returning a Limited Set of Dispute Data
PCM_OP_AR_GET_ACCT_BAL_SUMMARY	Retrieving a Balance Summary
PCM_OP_AR_GET_ACCT_BILLS	Getting a List of Bills Retrieving a List of Bills for a Bill Unit
PCM_OP_AR_GET_ACTION_ITEMS	Retrieving A/R Items That Apply to a Bill Unit
PCM_OP_AR_GET_BAL_SUMMARY	Retrieving a Balance Summary
PCM_OP_AR_GET_BILL_ITEMS	Getting Bill Items Retrieving a List of Bill Items for a Bill Unit

Table 4-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_AR_GET_BILLS	Getting a List of Bills Getting Bills Retrieving a List of Bills for a Bill Unit
PCM_OP_AR_GET_DISPUTE_DETAILS	Retrieving Dispute Details for a Bill Unit Retrieving Details on Available Balances
PCM_OP_AR_GET_DISPUTES	Retrieving Dispute Details for a Bill Unit
PCM_OP_AR_GET_ITEM_DETAIL	Retrieving Details about a Specific A/R Item or Bill Item Returning a Limited Set of Dispute Data
PCM_OP_AR_GET_ITEMS	Retrieving Details about a Specific A/R Item or Bill Item Retrieving Details on Available Balances
PCM_OP_AR_ITEM_ADJUSTMENT	Adjusting Items Transferring Balances between Items Adjusting Bills Tax Processing for Bill and Item Level Adjustments
PCM_OP_AR_ITEM_DISPUTE	Disputing Items Settling Disputed Bills Disputing Bills
PCM_OP_AR_ITEM_SETTLEMENT	Settling Disputed Items Settling Disputed Bills Transferring Balances between Items
PCM_OP_AR_ITEM_WRITEOFF	About Initiating Write-Offs Performing Write Offs About Bill Unit Write-Offs About Bill Write-Offs About Taxes for Write-Offs Customizing Write-Off Validation Transferring Balances between Items
PCM_OP_AR_POL_REVERSE_WRITEOFF	Reversing Write-Offs Customizing the Rules for Performing Write-Off Reversals
PCM_OP_AR_RESOURCE_AGGREGATION	Retrieving Details on Available Balances Retrieving Details about a Specific A/R Item or Bill Item Retrieving a Full Set of Dispute Data
PCM_OP_AR_REVERSE_REFUND	Reversing Refunds
PCM_OP_AR_REVERSE_WRITEOFF	Reversing Write-Offs About Initiating Write-Off Reversals Customizing the Rules for Performing Write-Off Reversals Transferring Balances between Items
PCM_OP_BAL_CHANGE_VALIDITY	Modifying the Sub-balance Validity Period
PCM_OP_BAL_GET_ACCT_BILLINFO	Finding a Bill Unit
PCM_OP_BAL_POL_SET_SUB_BALANCES	Customizing Balance Transfers
PCM_OP_BAL_TRANSFER_BALANCE	Transferring Balances between Accounts or Services

Table 4-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_DEBIT	Adjusting Accounts, Subscription Services, and Member Services Applying Debits and Credits
PCM_OP_BILL_FIND	Finding a Bill
PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT	Finding Discounts in Bill Items
PCM_OP_BILL_ITEM_EVENT_SEARCH	Finding Events Associated with Bill Items
PCM_OP_BILL_ITEM_REFUND	Transferring Balances between Items
PCM_OP_BILL_ITEM_TRANSFER	Reversing Refunds Transferring Balances between Items Disputing Events Settling Disputed Events
PCM_OP_BILL_POL_EVENT_SEARCH	Finding Events Associated with an Account
PCM_OP_BILL_POL_ITEM_VALID_SETTLEMENT	Settling Disputed Items
PCM_OP_BILL_POL_ITEM_VALID_WRITEOFF	Performing Write Offs
PCM_OP_BILL_POL_REVERSE_PAYMENT	Customizing Reversal of Payments Allocated to Written-Off Accounts About Initiating Write-Off Reversals Reversing a Write-Off Reversal
PCM_OP_BILL_POL_VALID_ADJUSTMENT	Adjusting Items Customizing Item Adjustments
PCM_OP_BILL_POL_VALID_DISPUTE	Disputing Items Customizing Item Disputes Customizing Item Settlements
PCM_OP_BILL_POL_VALID_SETTLEMENT	Disputing Items Customizing Item Settlements
PCM_OP_BILL_POL_VALID_TRANSFER	Customizing Item Transfer Validation
PCM_OP_BILL_POL_VALID_WRITEOFF	Customizing Write-Off Validation
PCM_OP_BILL_RCV_PAYMENT	Transferring Balances between Items
PCM_OP_BILL_REVERSE_PAYMENT	Reversing a Write-Off Reversal Customizing Reversal of Payments Allocated to Written-Off Accounts Transferring Balances between Items
PCM_OP_BILL_SET_LIMIT_AND_CR	Creating Balance Groups Modifying Sub-balances
PCM_OP_BILL_TAX_EVENT	Tax Processing for Bill and Item Level Adjustments Tax Processing for Disputes Tax Processing for Settlements

Table 4-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_TRANSFER	Adjusting Items Disputing Items Settling Disputed Items Performing Write Offs Customizing Reversal of Payments Allocated to Written-Off Accounts
PCM_OP_BILL_TRANSFER_BALANCE	Transferring Balances between Balance Groups
PCM_OP_GROUP_UPDATE_INHERITED	Updating Inheritance Fields in Groups
PCM_OP_IFW_SYNC_PUBLISH_EVENT	Transferring Services between Balance Groups
PCM_OP_PYMT_COLLECT	About Initiating Write-Off Reversals Reversing Write-Offs
PCM_OP_PYMT_ITEM_SEARCH	Finding Items
PCM_OP_SUBSCRIPTION_CANCEL_DEAL	Transferring Services between Balance Groups
PCM_OP_SUBSCRIPTION_PURCHASE_DEAL	Transferring Services between Balance Groups
PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS	Transferring Services between Balance Groups
PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER	Transferring Services between Balance Groups
PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER	Performing Rollover Transfers
PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION	Transferring Services between Balance Groups

Getting a List of A/R Items

PCM_OP_AR_GET_ACCT_ACTION_ITEMS retrieves the list of A/R items applied to all bill units (**/billinfo** objects) in an account or to a single bill unit.

You can restrict the search by various means, such as date, status, and bill unit POID.

You can find items for the specific bill unit or for it and its nonpaying child bill units.

PCM_OP_AR_GET_ACCT_ACTION_ITEMS uses the PIN_FLD_FLAGS both as an input and output field.

PCM_OP_AR_GET_ACCT_ACTION_ITEMS uses the input PIN_FLD_FLAGS value to select and/or mark the required items:

- When PIN_FLD_FLAGS contains PIN_AR_BILLED_ITEM, it selects the billed items that are allocated to the specified bill and marks each item as "billed."
- When PIN_FLD_FLAGS contains PIN_AR_UNBILLED_ITEM, it selects the unbilled items that are allocated to the specified bill and marks each item as "unbilled."
- If PIN_FLD_FLAGS is not present or is present but does not have either value, this opcode selects both billed and unbilled items allocated to the bill, but does not mark the A/R items.

If the PIN_FLD_FLAGS field is present, PCM_OP_AR_GET_ACCT_ACTION_ITEMS expects a value for the PIN_FLD_BILL_OBJ input field, or it returns an error.

PCM_OP_AR_GET_ACCT_ACTION_ITEMS determines that a special item is a “billed" item if it is allocated to a bill before the bill is finalized or before the corrective bill is created.

The PIN_FLD_RESULTS output array contains the PIN_FLD_FLAGS which indicates whether the allocated item is billed (PIN_AR_BILLED_ITEM) or unbilled (PIN_AR_UNBILLED_ITEM). This output array contains the PIN_FLD_FLAGS entry if the input flist contains PIN_FLD_FLAGS field and if a special item is allocated to the bill.

The special items that PCM_OP_AR_GET_ACCT_ACTION_ITEMS considers include all of the following:

- PIN_OBJ_TYPE_ITEM_PAYMENT
- PIN_OBJ_TYPE_ITEM_REVERSAL
- PIN_OBJ_TYPE_ITEM_REFUND
- PIN_OBJ_TYPE_ITEM_ADJUSTMENT
- PIN_OBJ_TYPE_ITEM_DISPUTE
- PIN_OBJ_TYPE_ITEM_SETTLEMENT
- PIN_OBJ_TYPE_ITEM_WRITEOFF
- PIN_OBJ_TYPE_ITEM_WRITEOFF_REVERSAL

Reversing Refunds

If a refund fails, you can reverse the refund. Refunds can fail because of returned checks and invalid credit cards.

- You can reverse only failed refunds.
- A refund can be reversed only once. You cannot reverse a refund that has been reversed before.

To reverse a refund, use PCM_OP_AR_REVERSE_REFUND.

PCM_OP_AR_REVERSE_REFUND does the following:

1. Takes as input one or more **/item/refund** POIDs in the input flist.
2. For each PIN_FLD_REVERSALS array, PCM_OP_AR_REVERSE_REFUND does the following:
   - Checks if the refund is already reversed

     Refunds that are already reversed cannot be reversed again.
   - Checks if the refund is a failed refund
3. Fetches the **/event/billing/transfer/item** event for each **/item/refund** in the input flist.

   The PIN_FLD_ACTION_INFO field of the **/event/billing/transfer/item** event contains the details of the **/item/refund** item from which the amount is transferred and the details of the **/item/refund** item to which the amount is transferred.
4. Creates the **/event/billing/refund/reversal** event that points to the **/item/refund** item in the input flist.

   The input does not contain the BALANCE_IMPACTS array. The PIN_FLD_REFUND substruct contains the amount that is being reversed.

   The SESSION_OBJ field of **/event/billing/refund/reversal** points to the corresponding **/event/billing/refund/cc** that failed.

5. Debits the refund amount from the **/item/refund** item and calls PCM_OP_BILL_ITEM_TRANSFER to transfer the refund amount back to the original items (for example, a credit amount to the **/item/payment** item).
6. Sets **/item/refund** to **0** and closes **/item/refund**.

   The PIN_FLD_SESSION_OBJ field of the transfer events point to **/event/billing/refund/reversal**.
7. Creates an **/event/billing/refund/reversal** event.

   The **/event/billing/refund/reversal** event does not contain the BALANCE_IMPACTS array.

Transferring Amounts between Items

During A/R operations, such as settlements and refunds, balances are transferred between items and between balance groups.

BRM uses the following opcodes for transfers:

- PCM_OP_BILL_ITEM_TRANSFER

  See "Transferring Balances between Items".
- PCM_OP_BILL_TRANSFER_BALANCE

  See "Transferring Balances between Balance Groups".
- PCM_OP_BILL_POL_VALID_TRANSFER

  See "Customizing Item Transfer Validation".

Transferring Balances between Items

To transfer balances between items, BRM uses PCM_OP_BILL_ITEM_TRANSFER. This opcode can accept items from multiple A/R bills and creates one transfer event for each A/R bill.

PCM_OP_BILL_ITEM_TRANSFER is called by the following opcodes:

- PCM_OP_BILL_RCV_PAYMENT
- PCM_OP_BILL_REVERSE_PAYMENT
- PCM_OP_AR_EVENT_ADJUSTMENT
- PCM_OP_AR_EVENT_DISPUTE
- PCM_OP_AR_EVENT_SETTLEMENT
- PCM_OP_AR_ITEM_ADJUSTMENT
- PCM_OP_AR_ITEM_DISPUTE
- PCM_OP_AR_ITEM_SETTLEMENT
- PCM_OP_AR_ITEM_WRITEOFF
- PCM_OP_AR_REVERSE_WRITEOFF
- PCM_OP_BILL_ITEM_REFUND

The POID of the source item is specified in the PIN_FLD_ITEM_OBJ field in the input flist. Each target item is specified in its own PIN_FLD_ITEMS array in the input flist. This array also includes the POID of the item's bill object and A/R bill object.

The POID of the corresponding A/R event is specified in the PIN_FLD_SESSION_OBJ field in the input flist and is stored in the PIN_FLD_SESSION_OBJ field of the transfer event. This associates the adjustment, dispute, or settlement event with the event that was used to transfer the A/R amount from an A/R item to the appropriate bill item.

> **Note:**
>
> If the transfer is not made as a result of an A/R operation, the value is the session in which the event took place; payment batch event, refund batch event, or reversal batch event.

Each call to PCM_OP_BILL_ITEM_TRANSFER makes a single transfer to an A/R bill. Each bill can include multiple target items. Amounts are allocated to items in the bill starting with the lowest-level items.

PCM_OP_BILL_ITEM_TRANSFER:

- Reads the target item
- Changes the target item's fields. The fields changed depend on the source item. For example, if the source item is an adjustment, the PIN_FLD_ADJUSTED field of the target item is changed.
- Changes the source item fields as follows:
  - PIN_FLD_DUE -= PIN_FLD_AMOUNT
  - PIN_FLD_TRANSFERED += PIN_FLD_AMOUNT
- If PCM_OP_BILL_ITEM_TRANSFER was called as part of settling a dispute, checks the PIN_FLD_DISPUTED field in the flist to determine the dispute amount.
  The PIN_FLD_DISPUTED field is only populated in the flist if PCM_OP_BILL_ITEM_TRANSFER is called by an item and event settlement, and it is used to prevent misdirection of settlement balances. It ensures proper application of the settlement in cases where a single item may be disputed by both an item dispute and an event dispute.

  For item settlements, the amount in this field is the dispute amount that originated solely from the item dispute. This amount does not include any contribution from an event dispute. For event settlements, the PIN_FLD_DISPUTED field provides the dispute amount that originated solely from the event dispute being settled.

  Using this information, PCM_OP_BILL_ITEM_TRANSFER populates the settled amount in the original item's PIN_FLD_ADJUSTED field to record the balance impact of the settlement. If any of the disputed amount was denied in the settlement, this opcode also increases the PIN_FLD_DUE field in the original item by the denied amount. These activities are summarized as follows:
  - PIN_FLD_DISPUTED -= PIN_FLD_DISPUTED
  - PIN_FLD_ADJUSTED += PIN_FLD_AMOUNT
  - PIN_FLD_DUE += (PIN_FLD_DISPUTED - PIN_FLD_AMOUNT)
- Writes the **/event/billing/item/transfer** event, which includes the following fields in addition to those it inherits:
  - PIN_FLD_BUFFER: List of all items or bills affected
  - PIN_FLD_AMOUNT: Amount of the transfer

  - PIN_FLD_AR_BILL_OBJECT: A/R bill object of the items in the transfer
  - PIN_FLD_SESSION_OBJ: The associated A/R event or batch event in which the A/R action took place (payment batch event, refund batch event, or reversal batch event.
- If (PIN_FLD_DUE == PIN_FLD_DISPUTED == 0) marks the status PIN_ITEM_STATUS_CLOSED.
- Writes the modified source and target items.
- Sets the bill's PIN_FLD_STATE field to one of the following:
  - **PARTIALLYPAID**: If the value of PIN_FLD_DUE for the bill is greater than 0 and less than the value of PIN_FLD_CURRENT_TOTAL.
  - **SETTLED**: If the value of PIN_FLD_DUE for the bill is equal to 0 and the value of PIN_FLD_CURRENT_TOTAL is greater than 0.

Transferring from a Negative to a Positive Balance

PCM_OP_AR_BILL_CREDIT_TRANSFER transfers the amount from a bill that has a negative balance to one or more bills that have a positive balance.

This opcode allocates credit amounts to bills that have a positive balance. This opcode takes as input the **/bill** object and corresponding **/billinfo** object of both the source bill and destination bill or bills.

Transferring Balances between Balance Groups

To transfer balances between balance groups, BRM uses PCM_OP_BILL_TRANSFER_BALANCE.

For example, BRM uses this opcode to transfer funds from one account to another.

PCM_OP_BILL_TRANSFER_BALANCE transfers an asset from a credit balance to another balance by debiting the source balance and crediting the target balance with the PIN_FLD_AMOUNT specified in the input flist. For example:

- Balance group A has a credit balance of $100 (represented for accounting purposes as **-100**).
- Balance group B has a credit balance of $2 (**-2**).
- If PIN_FLD_AMOUNT is **30**, PCM_OP_BILL_TRANSFER_BALANCE transfers $10 from balance group A to balance group B with these results:
  - Balance group A now has a credit balance of $70 (**-70**).
  - Balance group B now has a credit balance of $32 (**-32**).

If the PIN_FLD_VERIFY_BALANCE setting in the PCM_OP_BILL_TRANSFER_BALANCE input flist is set to PIN_BOOLEAN_FALSE, PCM_OP_BILL_TRANSFER_BALANCE can also transfer an asset from a debit balance. For example:

- Balance group A has a debit balance of $10 (represented as **+10**).
- Balance group B has a credit balance of $2 (**-2**).
- If PIN_FLD_AMOUNT is **30** and PIN_FLD_VERIFY_BALANCE is PIN_BOOLEAN_FALSE, PCM_OP_BILL_TRANSFER_BALANCE transfers $30 from balance group A to balance group B with these results:
  - Balance group A now has a debit balance of $40 (**+40**).

– Balance group B now has a credit balance of $32 (**-32**).

If the PIN_FLD_VERIFY_BALANCE field is not set (default) or is set to PIN_BOOLEAN_TRUE, PCM_OP_BILL_TRANSFER_BALANCE *cannot* transfer assets from post-paid balances.

If the PIN_FLD_AMOUNT field is greater than or equal to **0**, the transfer succeeds. If the PIN_FLD_AMOUNT field is less than **0**, the PIN_FLD_RESULT value in the output flist is set to **2**.

Table 4-2 lists the entries for the PIN_FLD_RESULT field:

Table 4-2 PIN_FLD_RESULT Values

Value	Meaning
0	Balance transfer was successful
1	Insufficient funds in the source account
2	Transfer amount was less than 0

These values are passed to PCM_OP_AR_ACCOUNT_ADJUSTMENT, which debits and credits the source and target accounts respectively.

> **Note:**
>
> When you use PCM_OP_BILL_TRANSFER_BALANCE, billing items are left unallocated as a result of the PCM_OP_AR_ACCOUNT_ADJUSTMENT calls.

Customizing Item Transfer Validation

To customize how to validate amounts being transferred, use PCM_OP_BILL_POL_VALID_TRANSFER.

Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

PCM_OP_BILL_POL_VALID_TRANSFER is called by PCM_OP_BILL_ITEM_TRANSFER.

Managing Currency and Non-Currency Balance Transfers between Accounts or Services

Use the following opcodes to transfer currency or non-currency balances from one account or service to another account or service:

- PCM_OP_BAL_TRANSFER_BALANCE

  See "Transferring Balances between Accounts or Services".
- PCM_OP_BAL_POL_SET_SUB_BALANCES

  See "Customizing Balance Transfers".

Transferring Balances between Accounts or Services

To transfer currency or non-currency balances from one account or service to another, use PCM_OP_BAL_TRANSFER_BALANCE. This opcode transfers balances regardless of the source's and target's balance type: postpaid, prepaid, or hybrid. For example, this opcode could transfer 200 free minutes from a prepaid balance to a postpaid balance.

This opcode is called by Billing Care, Customer Center, and the Billing Care REST API.

When the opcode is called, pass the following information in the opcode's input flist:

- The POID of the source account, service, or balance group you are transferring from. If you specify the POID of an account or service, its default balance group is used.
- The POID of the destination account, service, or balance group you are transferring to. If you specify the POID on an account or service, its default balance group is used.
- The transfer amount and resource ID.
- By default, the transfer balance keeps the source's validity period. To modify the validity period of the transfer balance's validity period, pass in the PIN_FLD_SUB_BALANCES array with the following input flist fields:
  - PIN_FLD_VALID_FROM: Set this to the timestamp of the validity's start date and time.
  - PIN_FLD_VALID_TO: Set this to the timestamp of the validity's end date and time.
  - PIN_FLD_ELEMENT_ID: Set this to the balance element ID.
- Whether to charge a transaction fee to the source, destination, both, or none. To do so, set the PIN_FLD_FEE_FLAG input flist field to **0** (do not apply a fee), **1** (apply a fee to the source), **2** (apply a fee to the destination), or **3** (apply a fee to both the source and destination). If charging a transaction fee, also include the flat amount or percentage of the balance transfer to apply:

  To apply a flat fee, set these input flist fields:
  - PIN_FLD_CHARGE_AMT: The amount of the transaction fee.
  - PIN_FLD_CHARGE_RESOURCE_ID: The currency or non-currency resource ID of the transaction fee, such as 840 for USD or 111055 for free minutes.

  To apply a percentage of the balance transfer amount, set the PIN_FLD_PERCENT field to the percentage, such as **3** for 3%.

After it receives the input flist, PCM_OP_BAL_TRANSFER_BALANCE performs the following tasks:

1. Opens a global transaction.
2. If an **/account** or **/service** POID is passed in, calls PCM_OP_BAL_GET_BALANCES to retrieve its default **/balance_group** POID and its current currency or non-currency balance.
3. Generates an audit event (**/event/audit/transfer_balance** object).
4. Calls PCM_OP_BAL_POL_SET_SUB_BALANCES to perform any customizations to the transfer balance being debited from the source.
5. Calls PCM_OP_BILL_DEBIT to debit the transfer amount from the source account, service, or balance group.
6. Calls PCM_OP_BAL_POL_SET_SUB_BALANCES to perform any customizations to the transfer balance being credited to the destination.

7. Calls PCM_OP_BILL_DEBIT to credit the transfer amount to the destination account, service, or balance group.
8. If there is a transfer fee, calls the PCM_OP_ACT_USAGE to generate a balance transfer event (**/event/billing/fee/balance_transfer** object).
9. Closes the global transaction.

Customizing Balance Transfers

To customize balances before they are debited from the source account, service, or balance group, or after they are credited to the destination account, service, or balance group, use the PCM_OP_BAL_POL_SET_SUB_BALANCES policy opcode. By default, this policy opcode does nothing.

This policy opcode is called by PCM_OP_BAL_TRANSFER_BALANCE.

Transferring Services between Balance Groups

Use PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER to transfer a service from one balance group to another in the same account or in a different account. You must create a custom application to implement this functionality.

When transferring services between balance groups of different accounts, be aware of the following limitations:

- The service's existing balance is transferred to the new balance group.
- Only one service can be transferred at a time, except for line services. For line service transfers, the entire line is transferred.
- Services cannot be transferred to an existing balance group.

When a service transfers to a new balance group in the same account, each balance group keeps its existing balance. Events and call detail records (CDRs) with a time stamp *before* the transfer continue to impact the old balance group, and events and CDRs with a time stamp *after* the transfer impact the new balance group.

For example, assume that an account owns two services: email and broadband access. Each is assigned its own balance group. On June 1, balance group A has a balance of $30, and balance group B has a balance of $20. On June 15, the account transfers the email service to balance group A. Balance group A still has a balance of $30, and balance group B still has a balance of $20 as shown in Table 4-3.

Table 4-3 Multiple Balance Groups Example

Balance Group	June 1	June 15
A Service	**/service/ip**	**/service/ip** **/service/email**
Balance	$30	$30
B Service	**/service/email**	None
Balance	$20	$20

In this example, all email-related charges that occur on or after June 15 impact balance group A, and those that occurred before June 15 impact balance group B.

Table 4-4 shows how transferring services between balance groups impacts other BRM features:

Table 4-4 Impact of Transferring Services between Balance Groups

Affected Feature	Description
Charge and discount sharing groups	The way transfers affect charge and discount sharing groups depends on whether the service is transferring to a balance group in the same account or in a different account: • **In the same account**: If the service being transferred is the owner of a charge or discount sharing group, the charges or discounts are shared by the new balance group. • **To a different account**: If the service being transferred is the owner of a charge or discount sharing group, the group is unlinked and you must create a new group. See "Managing Sharing Groups".
Delayed CDRs	The way transfers affect delayed CDRs depends on whether the event occurred *before* or *after* the service was transferred: • **Before**: Balance impacts are applied to the old balance group. • **After**: Balance impacts are applied to the new balance group.
Subscriptions	The way transfers affect subscriptions depends on whether you are transferring a subscription service or a member service: • **Subscription services**: BRM transfers all member services that share the same balance group to the new balance group. • **Member services**: BRM does not transfer the other member services in the subscription. **Note:** Subscription services and their member services must all point to the same balance group. See "Managing Customers' Subscription Services" in *BRM Managing Customers*.
Bill items	When a service transfers to a new balance group, BRM creates bill items for the new balance group. Events and CDRs with a time stamp *before* the transfer continue to impact the old bill items, and events and CDRs with a time stamp *after* the transfer impact the new bill items. See “About Bill Items" in *BRM Configuring and Running Billing*.
Backdating	After a service transfers to a new balance group, the service can no longer be backdated. Also, the service transfer cannot be backdated or future dated.

PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER stores data about a service's old and new balance groups and the time stamp of when the transfer occurred in the **/service** object's PIN_FLD_TRANSFER_LIST array.

When BRM applies balance impacts or retrieves balance group information, PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER automatically checks the **/service** object's PIN_FLD_TRANSFER_LIST array to determine which balance group it should use based on the event time stamp. If the array does not list a balance group, this opcode automatically uses the service's default balance group.

PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER is a wrapper opcode that calls other opcodes to perform the following:

- Transfer a service between balance groups in the same account or transfer services when an account purchases a bundle.
- Transfer services from the balance group of one account to the balance group of a different account.
- Reassign a balance group of one bill unit to another bill unit in the same account.

> **Note:**
>
> Use PCM_OP_CUST_MODIFY_CUSTOMER to reassign balance groups to a different bill unit because it performs additional validations during the reassignment process. See "Moving a Balance Group from One Bill Unit to Another".

PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER performs the following tasks:

1. Closes the service's accounting cycle and locks the balance group that the service is transferring from.
2. Generates the **event/notification/service_balgrp_transfer/start** notification event.

> **Note:**
>
> By default, this notification event does not trigger an action. However, you can use event notification to perform custom actions before the transfer occurs.

3. Determines whether the service is transferring to a new balance group by checking the PIN_FLD_TO_BAL_GRP_OBJ input flist field:
   - If the field contains a type-only POID, the service is transferring to a new balance group. The opcode calls PCM_OP_CUST_SET_BAL_GRP (see step 7).
   - If the field contains a complete POID, continue to the next step.
4. Determines whether a balance group is being reassigned to a different bill unit in the same account by comparing the PIN_FLD_TO_BAL_GRP_OBJ and PIN_FLD_FROM_BAL_GRP_OBJ fields.
   - If the fields are different, determines whether the service to be transferred is a subscription service (see step 5).
   - If the fields are the same and neither the PIN_FLD_BILLINFO_OBJ field nor the PIN_FLD_BILLINFO array are passed in, determines whether the service to be transferred is a subscription service (see step 5).
   - If the fields are the same and the PIN_FLD_BILLINFO_OBJ field is passed in, the balance group is being reassigned to an *existing* bill unit. PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER locks the balance group that the service is transferring to (see step 8).
   - If the fields are the same and the PIN_FLD_BILLINFO array is passed in, the balance group is being reassigned to a *new* bill unit. PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER creates the bill unit by:
     a. Calling PCM_OP_CUST_SET_PAYINFO to add the bill unit's payment information.
     b. Calling PCM_OP_CUST_SET_BILLINFO to update the bill unit's billing information.
     c. Calling PCM_OP_CUST_SET_BAL_GRP (see step 7).
5. Determines whether the service to be transferred is a subscription service.

   If it is, all member services must be transferred and reassigned. The opcode performs the following validations:

- If the transfer is across accounts, it validates that the member services were not passed in the input flist, that the subscription service and member services are assigned to the same balance group, and that the services passed in the input flist all point to PIN_FLD_FROM_BALANCE_GROUP.
- If the transfer is in the same account, it validates that the services passed in the input flist all point to PIN_FLD_FROM_BALANCE_GROUP.

6. Determines whether a new bundle was purchased by checking if the PIN_FLD_DEALS array was passed in the input flist for the **/service** object.

   If the array was passed in, a new bundle was purchased and all of the service's existing bundles must be canceled. PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER performs the following:

   a. Calls PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS to retrieve the account's hierarchical relationships of bundles, charge offers, discount offers, and services.

   b. Calls PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel all of the service's existing bundles and, if it is a subscription service, to cancel all of the member service bundles.

7. Calls PCM_OP_CUST_SET_BAL_GRP.

   PCM_OP_CUST_SET_BAL_GRP performs one of the following:

   - If the PIN_FLD_TO_BAL_GRP_OBJ input flist field contains a type-only POID, creates a new balance group and then associates the service with the new balance group.
   - If the PIN_FLD_TO_BAL_GRP_OBJ input flist field contains a complete POID, associates the service with the specified balance group.
   - If the PIN_FLD_TO_BAL_GRP_OBJ input flist field contains a complete POID and the PIN_FLD_BILLINFO_OBJ field or PIN_FLD_BILLINFO array is passed in, associates the balance group with the specified bill unit.

   > **Note:**
   >
   > If it is a subscription service, PCM_OP_CUST_SET_BAL_GRP is called for each member service as well.

8. Locks the balance group that the service is transferring to. PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER then does the following:

9. Determines whether the service that is being transferred is the owner of a sharing group.

   If it is, this opcode updates the **/group/sharing** object's PIN_FLD_BAL_GRP_OBJ field with the new balance group.

10. Determines whether the transfer is to a different account by checking the PIN_FLD_TO_BAL_GRP_OBJ and PIN_FLD_FROM_BAL_GRP_OBJ fields.

    If the fields are different, this opcode calls PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION to transfer the service and any member services from one account to another.

11. Determines whether a new bundle was purchased by checking if the PIN_FLD_DEALS array was passed in the input flist for the **/service** object.

    If the array was passed in, this opcode calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to purchase the new bundle.

12. Creates the **/event/audit/service_balgrp_transfer** audit event.

13. Generates the **/event/notification/service_balgrp_transfer/data** notification event. This triggers a call to PCM_OP_IFW_SYNC_PUBLISH_EVENT.
14. Generates the **/event/notification/service_balgrp_transfer/end** notification event.

> **Note:**
>
> By default, this notification event does not trigger an action. However, you can use event notification to perform custom actions after the transfer occurs.

Performing Adjustments

You use Billing Care or Customer Center to make adjustments. You can create a custom application to handle specialized adjustment scenarios, such as low-level adjustments against specific events or high-level adjustments distributed across an A/R account and its child accounts. For example, you might implement different adjustment levels to enable the CSR to help a customer who was charged too many minutes for a call and a customer whose prepaid service experienced a 3-day outage.

Use the following opcodes to perform and customize adjustments:

- PCM_OP_AR_ACCOUNT_ADJUSTMENT

  See "Adjusting Accounts, Subscription Services, and Member Services".
- PCM_OP_AR_BILL_ADJUSTMENT

  See "Adjusting Bills".
- PCM_OP_AR_ITEM_ADJUSTMENT

  See "Adjusting Items".
- PCM_OP_BILL_POL_VALID_ADJUSTMENT

  See "Customizing Item Adjustments".
- PCM_OP_AR_EVENT_ADJUSTMENT

  See "Adjusting Events".
- PCM_OP_ACT_POL_SPEC_GLID

  See "Assigning G/L IDs for an Adjustment".

Adjusting Accounts, Subscription Services, and Member Services

To adjust accounts, subscription services, or member services, BRM uses PCM_OP_AR_ACCOUNT_ADJUSTMENT. This opcode debits or credits the currency balance for the specified account. This opcode is called by BRM client applications to adjust the balance impacts of an event.

If the input flist specifies a balance group, BRM debits or credits that **/balance_group** object. If the input flist specifies a bill unit, the adjustment is applied to the default balance group of the bill unit. If the input flist specifies account only, it debits or credits the account default balance group.

You can open an account adjustment with or without tax. For information on how PCM_OP_AR_ACCOUNT_ADJUSTMENT performs adjustments taxation, see "Including Taxes in the Adjustment".

The **/event/billing/adjustment/account** object records the balance impact. The **item_total** and **due** fields of the adjustment item show the amount of adjustment. The adjustment item is billed immediately after the transaction occurs. The status of the item is set to PIN_ITEM_STATUS_OPEN.

If a balance group is not specified in the input flist, PCM_OP_AR_ACCOUNT_ADJUSTMENT adjusts the account default balance group. If the input flist identifies a balance group, PCM_OP_AR_ACCOUNT_ADJUSTMENT updates the matching sub-balance or creates a new sub-balance if it cannot find a match. The sub-balance that must be updated is also specified in the input flist.

If your client application enables you to specify a subscription service and a member service in the input flist, you can perform *subscription service adjustments* and *member service adjustments*. These adjustment levels use PCM_OP_AR_ACCOUNT_ADJUSTMENT to perform adjustments against all the services that belong to the balance group, which the **/service** object supplies.

If an account, subscription service, or member service adjustment is for noncurrency balances, use PCM_OP_BILL_DEBIT to adjust the noncurrency balance. For more information on how PCM_OP_BILL_DEBIT works with balances, see "Applying Debits and Credits".

BRM creates a G/L ID for every account adjustment. For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment".

Fields You Should Include in the Flist for Account, Subscription Service, and Member Service Adjustments

When performing account, subscription service, and member service adjustments, set these fields in the PCM_OP_AR_ACCOUNT_ADJUSTMENT input flist:

- PIN_FLD_POID for the account to which the adjustment applies. You obtain this value from the **/account** object that the CSR selects. Account, subscription service, and member service adjustments require this field.
- PIN_FLD_BILLINFO_OBJ for the bill unit to which the adjustment applies. The adjustment balance impact is applied to the default balance group of the bill unit.
- PIN_FLD_BAL_GRP_OBJ for the balance group to which the adjustment applies. You obtain this value from the **/account** or **/service** object that the CSR selects.

  This field is used for account, subscription service, and member service adjustments. If you omit this field from the input flist for an account adjustment, the adjustment uses the default balance group of the bill unit specified. Otherwise, it uses the account default balance group. Subscription service and member service adjustments require PIN_FLD_BAL_GRP_OBJ in the input flist.
- PIN_FLD_AMOUNT for the adjustment amount. The value supplied for this field should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the balance to adjust.
- PIN_FLD_FLAGS for the tax treatment:
  - PIN_AR_WITHOUT_TAX: The account adjustment is nontaxable.
  - If the PIN_AR_WITH_TAX: The account adjustment is taxable.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, f**m_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The adjustment is nontaxable if the **pin.conf** entry is absent.

PCM_OP_AR_ACCOUNT_ADJUSTMENT accepts a variety of other fields used to define tax treatment, adjustment time, adjustment description, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".

Flags for Account, Subscription Service, and Member Service Adjustments

Use these flags with PCM_OP_AR_ACCOUNT_ADJUSTMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, all the fields in the event object are returned in addition to the POID.
- If the PCM_OPFLG_CALC_ONLY flag is set, no fields in the database are changed and the event object is not created. The fields that would have been used to create the event object and adjustment item are returned to the caller.
- If the PCM_OPFLG_CALC_ONLY flag is not set, the **/event/billing/adjustment/account** object is created to record details of the operation.

Adjusting Bills

To make adjustments at the bill level, BRM uses PCM_OP_AR_BILL_ADJUSTMENT. This opcode is called by BRM client applications to adjust the balance impacts of items associated with the specified bill.

PCM_OP_AR_BILL_ADJUSTMENT debits or credits a currency balance for the specified **/bill** object. You can open a bill adjustment with or without tax. For information on how PCM_OP_AR_BILL_ADJUSTMENT performs adjustment taxation, see "Including Taxes in the Adjustment".

> **Note:**
>
> PCM_OP_AR_BILL_ADJUSTMENT does not check to determine whether the item you are adjusting is billed yet. However, using this opcode to adjust pending items is not recommended. If you try to adjust pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any adjustments against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_BILL_ADJUSTMENT *does not* make adjustments when:

- The specified bill is not an A/R bill (for example, a bill from a nonpaying bill unit).
- The amount of the adjustment exceeds the total amount of the bill. For example, if the bill due is $5, you cannot perform a credit adjustment for $6. Similarly, if the bill due -$5, you cannot perform a debit adjustment for $6.

Bill adjustments can be performed against the entire bill or just specific bill items. When it receives an input flist, PCM_OP_AR_BILL_ADJUSTMENT creates a single, umbrella adjustment item for all bill items that require adjustment, as follows:

- If the input flist specifies individual bill items, the adjustment item is associated with each of these bill items. You must specify the adjustment amount for each bill item in the flist.
- If the flist does not specify any bill items, all bill items are eligible for adjustment. PCM_OP_AR_BILL_ADJUSTMENT opens adjustments for as many bill items as it can before it consumes the adjustment amount or percentage. In this case, the adjustment item covers only those bill items that PCM_OP_AR_BILL_ADJUSTMENT was able to fully or partially adjust before using up the adjustment amount or, if the adjustment is defined as a percentage of the bill, the specified percentage.

In either case, PCM_OP_AR_BILL_ADJUSTMENT passes the POID of the adjustment item and the appropriate adjustment amount to PCM_OP_AR_ITEM_ADJUSTMENT, which performs the adjustment for each of the associated bill items. An **/event/billing/adjustment/item** object is created for each adjusted item in the bill, recording the balance impact. The **due** fields of the adjustment items become **0** after the adjustment and the status of the item is set to PIN_ITEM_STATUS_CLOSED. For details on how PCM_OP_AR_ITEM_ADJUSTMENT works, see "Adjusting Items".

Before passing the adjustment item and amount to PCM_OP_AR_ITEM_ADJUSTMENT, PCM_OP_AR_BILL_ADJUSTMENT determines whether the adjustment has tax implications. If so, it checks the original **/item** object to determine the taxable portion of the item:

- If the item object stores only an amount generated by a tax event, the opcode omits it from tax calculation.
- If the item object stores only an amount generated by a usage event, the opcode calculates taxes for the amount.
- If the item object stores an amount generated by a tax event and an amount generated by a usage event, the opcode calculates the adjustment tax reversal only for the usage amount and not the tax amount.

For each item that can be adjusted, the opcode checks the events that make up the item to determine whether any of the events are tax exempt. If so, it omits these events from the amount it uses when calculating the adjustment tax reversal. As a result of eliminating any nontaxable components of the original item, the adjustment tax is calculated proportionally to the actual taxable amount of the item.

If the adjustment fails, PCM_OP_AR_BILL_ADJUSTMENT returns the reason in the PIN_FLD_DESCR field of the PIN_FLD_RESULTS array on the output flist.

BRM creates a G/L ID for every bill adjustment. For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment".

Fields You Should Include in the Flist for Bill Adjustments

When performing bill adjustments, set the following fields in the PCM_OP_AR_BILL_ADJUSTMENT input flist:

- PIN_FLD_POID for the bill to which the adjustment applies. You obtain this from the **/bill** object that the CSR identifies. The adjusted amount is applied to the default balance group of the bill unit associated with the bill.
- PIN_FLD_POID for the items to be adjusted under the PIN_FLD_ITEMS array. This array is required only to adjust a particular set of bill items in the bill.

> **Note:**
>
> This field must be at level 1 of the input flist for the item adjustment to work.

- PIN_FLD_AMOUNT for the adjustment amount. The value supplied for this field should be negative when crediting and positive when debiting.

  You can specify the adjustment amount through either PIN_FLD_AMOUNT or PIN_FLD_PERCENT. The PIN_FLD_AMOUNT field is mandatory and the PIN_FLD_PERCENT field is optional. If the input flist includes both fields, BRM uses the value in PIN_FLD_PERCENT and ignores the value in PIN_FLD_AMOUNT.

  You can provide the PIN_FLD_AMOUNT field for the bill itself or, if the adjustment is against an array of bill items, for the individual bill items. If you are performing an adjustment that credits the bill, the absolute value of PIN_FLD_AMOUNT must be greater than 0 and less than the total bill amount and its sign should be negative.
- PIN_FLD_FLAGS for the tax treatment:
  - PIN_AR_WITHOUT_TAX: The bill adjustment is nontaxable.
  - If the PIN_AR_WITH_TAX: The bill adjustment is taxable.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, f**m_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The adjustment is nontaxable if the **pin.conf** entry is absent.

PCM_OP_AR_BILL_ADJUSTMENT accepts a variety of other fields used to define tax treatment, adjustment time, adjustment description, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".

BRM creates a G/L ID for every bill adjustment. For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment".

Flags for Bill Adjustments

Use these flags with PCM_OP_AR_BILL_ADJUSTMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, all the fields in the event object are returned in addition to the POID.
- If the PCM_OPFLG_CALC_ONLY flag is set, no fields in the database are changed and the event object is not created. The fields that would have been used to create the event object and adjustment item are returned to the caller.

If the PCM_OPFLG_CALC_ONLY flag is not set, the **/event/billing/adjustment/item** object is created to record details of the operation.

Adjusting Items

To adjust items, BRM uses PCM_OP_AR_ITEM_ADJUSTMENT. This opcode debits or credits a currency balance for the bill item specified in the input flist. The item adjustment balance impact is applied to the default balance group of the bill unit that is associated with the bill item. This opcode is also called by PCM_OP_AR_BILL_ADJUSTMENT to adjust items in a bill.

> **Note:**
>
> PCM_OP_AR_ITEM_ADJUSTMENT does not check to determine whether the item you are adjusting is billed yet. However, using this opcode to adjust pending items is not recommended. If you try to adjust pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any adjustments against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_ITEM_ADJUSTMENT does the following:

1. Reads the original **/item** object and prepares the changes to reflect the balance impact of the adjustment.

   For credit adjustments, these changes reduce the Due amount (PIN_FLD_DUE) for the item. For debit adjustments, the opposite occurs.

2. Calls PCM_OP_BILL_POL_VALID_ADJUSTMENT to validate the changes.

   You can customize the validation criteria for PCM_OP_BILL_POL_VALID_ADJUSTMENT. For information on customizing the opcode, see "Customizing Item Adjustments".

3. Checks to see if the PIN_FLD_ITEM_OBJ field is present in the input flist.

   If so, PCM_OP_AR_ITEM_ADJUSTMENT checks the contents of the field to determine:

   - Whether the field contains a POID, indicating that the opcode is being called by PCM_OP_AR_BILL_ADJUSTMENT to complete a bill adjustment.

     This POID identifies the **/item/adjustment** object that PCM_OP_AR_BILL_ADJUSTMENT created to store information on the bill adjustment. This is the **/item/adjustment** object that PCM_OP_AR_ITEM_ADJUSTMENT associates with each of the adjustment events it creates for the bill adjustment.

   - Whether the field is NULL, indicating that the opcode is being called directly from the client application to perform an item adjustment. In this case, PCM_OP_AR_ITEM_ADJUSTMENT creates its own **/item/adjustment** object. This object is exclusive to the individual item being adjusted.

4. If the adjustment has tax implications and PCM_OP_AR_ITEM_ADJUSTMENT is not being called as part of a bill adjustment, PCM_OP_AR_ITEM_ADJUSTMENT checks the original **/item** object to determine the taxable portion of the item:
   - If the item object stores only an amount generated by a tax event, the opcode omits it from tax calculation.
   - If the item object stores only an amount generated by a usage event, the opcode calculates taxes for the amount.
   - If the item object stores an amount generated by a tax event and an amount generated by a usage event, the opcode calculates the adjustment tax reversal only for the usage amount and not the tax amount.

   For each item that it can adjust, PCM_OP_AR_ITEM_ADJUSTMENT checks the events that make up the item to determine whether any of the events are tax exempt. If so, it omits these events from the amount it uses when calculating the adjustment tax reversal.

   As a result of eliminating any nontaxable components of the original item, the adjustment tax is calculated proportionally to the actual taxable amount of the item.

   > **Note:**
   >
   > If the item adjustment is being performed as part of a bill adjustment, PCM_OP_AR_BILL_ADJUSTMENT performs this step before calling PCM_OP_AR_ITEM_ADJUSTMENT.

5. Creates an **/event/billing/adjustment/item** object.

   If the item adjustment is part of a bill adjustment, the event object for each adjusted item is associated with a single, umbrella **/item/adjustment** object.

6. Examines the flags in the input flist to determine whether the adjustment has any tax implications.

   For information on how PCM_OP_AR_ITEM_ADJUSTMENT performs adjustment taxation, see "Including Taxes in the Adjustment".

7. Calls PCM_OP_BILL_TRANSFER to transfer funds to the A/R bill.

8. If the Due amount after adjustment is **0** and there are no open disputes, PCM_OP_AR_ITEM_ADJUSTMENT sets the PIN_ITEM_STATUS field to CLOSED.

   If an over-adjustment is made on an item, the following occurs:
   - For deferred taxation, the adjustment amount is left unallocated in the adjustment item. If the tax to be adjusted is greater than the cycle tax amount, then the cycle tax item is adjusted for its due amount.
   - For tax now taxation, the excess adjustment amount is left unallocated in the adjustment item.

9. Writes the modified version of the original **/item** object.

10. Provides feedback on whether the adjustment was successfully created.

    If the adjustment creation failed, it indicates the reason for the failure.

BRM creates a G/L ID for item adjustments. For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment".

Fields You Should Include in the Input Flist for Item Adjustments

When performing item adjustments, set these fields in the input flist:

- PIN_FLD_POID for the item to be adjusted. The adjusted amount is applied to the default balance group of the bill unit associated with the item.
- PIN_FLD_AMOUNT for the adjustment amount. The value should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the balance to adjust.

PCM_OP_AR_ITEM_ADJUSTMENT accepts a variety of other fields used to define tax treatment, adjustment time, adjustment description, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".

Flags for Item Adjustments

Use these flags with PCM_OP_AR_ITEM_ADJUSTMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_ITEM_ADJUSTMENT returns not only the POID, but all the fields in the **/event/billing/adjustment/item** object as well.
- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_ITEM_ADJUSTMENT does not change any fields in the database and does not create an **/event/billing/adjustment/item** object. However, it does provide an adjustment calculation to the caller by returning the fields that would have been used to create the event object and adjustment item.

> **Note:**
>
> If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_ITEM_ADJUSTMENT behaves in a standard manner, creating an **/event/billing/adjustment/item** object to record details of the operation.

- PIN_FLD_FLAGS for the tax treatment:
  - PIN_AR_WITHOUT_TAX flag: The item adjustment is nontaxable.
  - PIN_AR_WITH_TAX flag: The item adjustment is taxable. Tax is reversed on an item-by-item basis.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, f**m_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The adjustment is nontaxable if the **pin.conf** entry is absent.

Customizing Item Adjustments

PCM_OP_BILL_POL_VALID_ADJUSTMENT validates information to make adjustments against an item.

This opcode is called by PCM_OP_AR_ITEM_ADJUSTMENT and PCM_OP_AR_ACCOUNT_ADJUSTMENT.

When you initiate a bill or item adjustment, PCM_OP_AR_ITEM_ADJUSTMENT calls PCM_OP_BILL_POL_VALID_ADJUSTMENT to validate all proposed changes to the fields in the original **/item** object. This opcode checks the validity of field values before processing. Field values either pass or fail. If one field fails, the entire operation fails.

To customize the adjustment validation criteria, use PCM_OP_BILL_POL_VALID_ADJUSTMENT. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

Adjusting Events

To adjust events, BRM uses PCM_OP_AR_EVENT_ADJUSTMENT. This opcode debits or credits a balance for a specified event.

PCM_OP_AR_EVENT_ADJUSTMENT receives a list of events to be adjusted as a batch on the input flist, and returns the adjusted **/event/billing/adjustment/event** data on the output flist. Events that can be adjusted are stored in the **/config/adjustment/event** object. There can be multiple events in a given batch, and the events to be adjusted do not need to belong to the same account specified in the PIN_FLD_POID for PCM_OP_AR_EVENT_ADJUSTMENT. Events that can be adjusted are defined in the **init_objects.source** file and stored in the **/config/adjustment/event** object.

If you are adjusting pending items, you can ensure proper recording of the adjustment in the General Ledger by using PCM_OP_AR_EVENT_ADJUSTMENT to create a shadow adjustment rather than the standard event adjustment you would normally want to create for a billed item. For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment" and "Categorizing Unbilled Event Adjustments in G/L Reports".

By default, the adjustable events are:

- **/event/billing/debit**
- **/event/billing/product/fee/cycle/cycle_arrear**
- **/event/session/dialup**
- **/event/billing/product/fee/purchase**
- **/event/billing/product/fee/cancel**
- **/event/billing/product/fee/cycle/cycle_forward_monthly**
- **/event/billing/product/fee/cycle/cycle_forward_bimonthly**
- **/event/billing/product/fee/cycle/cycle_forward_quarterly**
- **/event/billing/product/fee/cycle/cycle_forward_semiannual**
- **/event/billing/product/fee/cycle/cycle_forward_annual**
- **/event/billing/product/fee/cycle/cycle_forward_arrear**

If you create custom events that you must adjust or if you must adjust additional events, use PCM_OP_WRITE_FLDS to add additional event types to the **/config/adjustment/event** object.

When PCM_OP_AR_EVENT_ADJUSTMENT receives a batch of events, it:

1. Identifies all events in the batch to be adjusted, and runs the validation checks against those events.

2. If the event has any tax implications, the tax implications must be adjusted.

   For information on how PCM_OP_AR_EVENT_ADJUSTMENT performs adjustment taxation, see "Including Taxes in the Adjustment".

3. Creates a single **/item/adjustment** object for all the events in the batch for the account.

4. For each event to be adjusted for an account, a new **/event/billing/adjustment/event** object containing the amount to be adjusted is created.

   The **/event/billing/adjustment/event** object has the PIN_FLD_ITEM_OBJ pointing to the adjustment item created in step 3.

5. After the adjustment item and the **/event/billing/adjustment/event** object are created, the amount is transferred from the adjustment item to the original item where the adjusted event belongs.

   The original item may or may not be open. If the transfer happens to a closed item, the closed item is reopened automatically.

> **Note:**
>
> After the entire Due amount from the adjustment item is transferred to other items, the adjustment item is closed.

BRM creates a G/L ID for event adjustments.

For information on adjustments and G/L IDs, see "Assigning G/L IDs for an Adjustment" and "Categorizing Unbilled Event Adjustments in G/L Reports".

Fields You Should Include in the Input Flist for Event Adjustments

When performing adjustment, set these fields in the PCM_OP_AR_EVENT_ADJUSTMENT input flist:

- PIN_FLD_POID for each of the events to be adjusted. The adjusted amount is applied to the default balance group of the bill unit associated with the **/item** object that the event belongs to.
- PIN_FLD_AMOUNT or PIN_FLD_PERCENT for the adjustment amount. The value supplied for this field should be positive for adjustments that credit the event and negative for adjustments that debit the event. If you omit both of these fields, BRM adjusts the entire amount for each event.
- PIN_FLD_RESOURCE_ID for the balance to adjust.

PCM_OP_AR_EVENT_ADJUSTMENT accepts a variety of other fields used to define tax treatment, adjustment time, adjustment description, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".

Flags for Event Adjustments

Use these flags with PCM_OP_AR_EVENT_ADJUSTMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_EVENT_ADJUSTMENT returns not only the POID, but all the fields in the **/event/billing/adjustment/event** object as well.

- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_EVENT_ADJUSTMENT does not change any fields in the database and does not create an **/event/billing/adjustment/event** object. However, it does provide an adjustment calculation to the caller by returning the fields that would have been used to create the event object and adjustment item.

  > **Note:**
  >
  > If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_EVENT_ADJUSTMENT behaves in a standard manner, creating an **/event/billing/adjustment/event** object to record details of the operation.

- If the PIN_ADJ_NO_TAX flag is set, the event adjustment is nontaxable. If this flag is not set, the adjustment is taxable.
- If the PIN_ADJ_TAX_ONLY flag is set, only the tax on the original event is adjusted; not the event itself.

  > **Note:**
  >
  > Do not set this flag for adjustments against unbilled events that use deferred taxation.

- If the PIN_EVENT_ADJ_BATCH is set, the adjustment amount is distributed among the events in the batch on a first-come-first-serve basis until the adjustment amount is consumed.
- If the PIN_EVENT_ADJ_EVENT is set, the adjustment amount is applied individually to each of the events in the batch.
- If the PIN_EVENT_ADJ_FOR_RERATE flag is set, the adjustment is the result of rerating the original event.
- If the PIN_AR_RECORD_SHADOW_ADJUSTMENT_EVENT is set, BRM creates a shadow adjustment instead of an **/event/billing/adjustment/event** object. This flag should be set when the event being adjusted is pending and results in the adjustment being recorded as unbilled in the General Ledger.

Categorizing Unbilled Event Adjustments in G/L Reports

To report adjustments as unbilled in the G/L, BRM uses PCM_OP_AR_EVENT_ADJUSTMENT to create shadow event objects instead of adjustment event objects. A *shadow event* object is a new version of the original event object. It is identical to the original event object except for a new creation timestamp and a new balance impact resulting from the event adjustment.

To create shadow adjustments, set the value of the PIN_FLD_FLAGS field at the top level to **1** (PIN_AR_RECORD_SHADOW_ADJUSTMENT_EVENT) on the PCM_OP_AR_EVENT_ADJUSTMENT input flist. The shadow object is created only if the original event to be adjusted has not been billed or posted to a G/L report.

> **Note:**
>
> When the PIN_FLD_FLAGS field is not present in the input flist, BRM creates an **/event/billing/adjustment/event** object.

Unlike standard event adjustments, when creating shadow events, only one event should be present on the input flist.

As with standard event adjustments, you can adjust both currency and noncurrency balances. Regardless of whether PCM_OP_AR_EVENT_ADJUSTMENT is performing a standard or shadow adjustment, it adjusts only the balances specified in the input flist; it does not automatically adjust all balances for an event.

For information on the shadow adjustment flags, see "Flags for Event Adjustments".

Including Taxes in the Adjustment

You can configure tax calculation on adjustments, as described in "Calculating Taxes for Accounts Receivable Actions" in *BRM Calculating Taxes*. You can specify an optional tax treatment for adjustments by using flags in the PCM_OP_AR_EVENT_ADJUSTMENT input flist. See the following topics:

- Flags for Account, Subscription Service, and Member Service Adjustments
- Flags for Bill Adjustments
- Flags for Item Adjustments
- Flags for Event Adjustments

Tax Processing for Account Adjustments

If the PCM_OP_AR_EVENT_ADJUSTMENT input flist flags the adjustment as taxable or the **pin.conf** file specifies that BRM tax all adjustments, PCM_OP_AR_EVENT_ADJUSTMENT obtains tax information from a configuration file read into BRM when Connection Manager starts.

PCM_OP_AR_ACCOUNT_ADJUSTMENT uses this information as follows:

- If adjustment tax method is tax now, it uses this information to calculate the tax and apply it immediately.
- If adjustment tax method is tax deferred, it enriches the input flist with this information, and then passes it to PCM_OP_ACT_USAGE for deferred tax calculation.

Tax Processing for Bill and Item Level Adjustments

If the PCM_OP_AR_EVENT_ADJUSTMENT input flist flags the adjustment as taxable or the **pin.conf** file specifies that BRM tax all adjustments, PCM_OP_AR_EVENT_ADJUSTMENT obtains tax information from a configuration file read into BRM when Connection Manager starts.

PCM_OP_AR_ITEM_ADJUSTMENT and PCM_OP_AR_BILL_ADJUSTMENT perform one of two steps:

- If adjustment tax method is tax now, it uses information from the event associated with the original bill item to calculate taxes and apply the amount immediately with the adjustment amount.
- If adjustment tax method is tax deferred, The taxes are contained in the tax item. In such cases, PCM_OP_AR_ITEM_ADJUSTMENT does the following to handle the taxes:
  - Calls PCM_OP_BILL_TAX_EVENT to calculate the tax amount.
  - Compares the calculated tax with the due amount of the tax item to determine the amount to reverse.
  - Creates the **/event/billing/adjustment/tax_event** event to contain the tax amount.
  - Transfers the calculated tax amount to the original tax item.

The due amount is reduced by the adjustment amount.

Validations are made to ensure that an item's net amount and cycle tax amount are not over-adjusted during an adjustment. If the adjustment amount is greater than the item's due amount, the excess amount is left unallocated in the adjustment item and must be allocated either manually or when the next payment is applied on the account. The tax amount is always adjusted to the maximum amount due in the tax item.

> **Note:**
>
> The adjustment tax method (tax now or tax deferred) is determined based on whether the events associated with the item are taxed at event time or at bill time.

Tax Processing for Event Adjustments

If the PCM_OP_AR_EVENT_ADJUSTMENT input flist flags the adjustment as taxable, PCM_OP_AR_EVENT_ADJUSTMENT performs one of two steps:

- If adjustment tax method is tax now, the opcode uses information from the adjusted event to calculate the tax and apply it immediately.
- If adjustment tax method is tax deferred, the opcode uses information from the **/event/billing/cycle/tax** object to calculate the adjustment tax. When it has calculated the tax, it creates the **/event/billing/adjustment/tax_event** event.

> **Note:**
>
> If the adjustment is created in the same billing cycle as the adjusted item, the original item will not have been taxed yet. In this case, there is no need to create the **/event/billing/adjustment/tax_event** event because the adjustment precedes initial tax calculation for the original item.

Including Reason Codes in the Adjustment

Reason codes explain why an adjustment is being granted. In typical implementations, the BRM client application provides a selectable set of reasons appropriate to the event type. You choose a reason that matches the situation or, in some cases, compose a new reason. The client application populates the input flist for the opcode with this information, and BRM records the reason in the adjustment event object (for example, **/event/billing/adjustment/item** or **/**

event/billing/adjustment/event). The flist fields that provide reason code information are PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID. These fields are optional.

For event adjustments, you can filter a batch of adjustments based on the reason code supplied in the input flist. If the PCM_OP_AR_EVENT_ADJUSTMENT input flist includes a reason code and domain, PCM_OP_AR_EVENT_ADJUSTMENT performs a validation check of the reason code. It does so by comparing the reason code in its input flist against a stored global flist created by reading reason code configuration information from the **/config/reason_code_scope** object at Connection Manager (CM) start time. The global flist serves as a filter that indicates all the reason codes that are valid for a given event and service type.

Reason code validation produces these results:

- For all instances where the reason code is valid for the event and associated service, the opcode proceeds with the adjustment.
- For each instance where the reason code is not valid, the opcode indicates the failure in the PIN_FLD_RESULTS field for the output flist and describes the problem in the PIN_FLD_DESCR field.

For general information about reason codes, see String Manipulation Functions in *BRM Developer's Reference*.

Assigning G/L IDs for an Adjustment

The G/L ID for an adjustment event is assigned like other pre-rated events in the BRM system. PCM_OP_ACT_POL_SPEC_GLID assigns the G/L ID for the adjustment. Based on the event type, this opcode retrieves a G/L ID from the **/config/map_glid** object. The PIN_FLD_GL_ID field in the **/event/billing/adjustment/event** object points to the same G/L ID as the original event, ensuring that both the original event and the adjustment are correctly recorded in the general ledger.

Applying Debits and Credits

To apply debits and credits to noncurrency balances, BRM uses PCM_OP_BILL_DEBIT.

PCM_OP_BILL_DEBIT debits sub-balances for a specific **/balance_group** object associated with an account.

There are three ways to use PCM_OP_BILL_DEBIT:

- Pass in the POID of an account to impact that account's default **/balance_group** object.
- Pass in the POID of a specific balance group to impact that **/balance_group** object.
- For either the account or a balance group, pass in a specific sub-balance to impact that sub-balance.

The PIN_FLD_DEBIT array on the input object contains the amount of the balance to debit. The amount is a signed value, so the result can be a debit (positive value) or a credit (negative value) to the balance.

The PIN_FLD_SUB_BALANCES array contains a specific element ID, or uses PIN_ELEMID_ASSIGN to identify or create the specific sub-balance to receive the debit.

You specify validity information using the VALID_FROM and VALID_TO fields on the PIN_FLD_SUB_BALANCES array to credit or debit a sub-balance for specific dates. These are usually used with the PIN_ELEMID_ASSIGN field to specify a sub-balance to credit or debit.

> **Note:**
>
> If you are performing a noncurrency account or service adjustment, PCM_OP_BILL_DEBIT determines whether the sub-balance being adjusted is still open. Open sub-balances are those whose VALID_TO date has not yet been reached. If the sub-balance is no longer open, the opcode creates a new sub-balance for the adjustment. This sub-balance has the same validity period as the sub-balance you were trying to adjust.

If validity information is not specified, PCM_OP_BILL_DEBIT updates the "most valid" sub-balance first, and then continues through the rest of the sub-balances until the debit amount is used up. If no other sub-balances are left, or if there were no valid sub-balances to begin with, PCM_OP_BILL_DEBIT it creates a new sub-balance to hold the debit amount.

> **Note:**
>
> - You can change the order in which PCM_OP_BILL_DEBIT updates sub-balances.
> - PCM_OP_BILL_DEBIT opens a local transaction.

Flags for Applying Debits and Credits

- If the PCM_OPFLG_READ_RESULT flag is set, all the fields in the event object are returned and not just the POID.
- If the PCM_OPFLG_CALC_ONLY flag is set, no fields in the database are changed and the event object is not actually created. The fields that would have been used to create the event object are returned to the caller.
- If the PCM_OPFLG_CALC_ONLY flag is not set, the **/event/billing/debit** object is created to record the details of the operation.

Performing Disputes and Settlements

You use Billing Care or Customer Center to manage disputes and settlements. BRM uses the following opcodes to perform and customize disputes and settlements:

- PCM_OP_AR_BILL_DISPUTE

  See "Disputing Bills".
- PCM_OP_AR_BILL_SETTLEMENT

  See "Settling Disputed Bills".
- PCM_OP_AR_ITEM_DISPUTE

  See "Disputing Items".
- PCM_OP_AR_ITEM_SETTLEMENT

  See "Settling Disputed Items".
- PCM_OP_BILL_POL_VALID_DISPUTE

See "Customizing Item Disputes".

- PCM_OP_AR_EVENT_DISPUTE

  See "Disputing Events".

- PCM_OP_AR_EVENT_SETTLEMENT

  See "Settling Disputed Events".

- PCM_OP_ACT_POL_SPEC_GLID

  See "Assigning G/L IDs for a Dispute or Settlement".

Disputing Bills

To open a dispute at the bill level, BRM uses PCM_OP_AR_BILL_DISPUTE. This opcode debits or credits a currency balance for the specified **/bill** object. You can open a bill dispute with or without tax. This opcode is called by BRM client applications to dispute the items associated with the specified bill. For information on how PCM_OP_AR_BILL_DISPUTE performs dispute taxation, see "Including Taxes in the Dispute or Settlement".

> **Note:**
>
> PCM_OP_AR_BILL_DISPUTE does not check to determine whether the item you are disputing is billed yet. However, using PCM_OP_AR_BILL_DISPUTE to dispute pending items is not recommended. If you try to dispute pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any disputes against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_BILL_DISPUTE *does not* create disputes when:

- The specified bill is not an A/R bill (for example, a bill from a nonpaying bill unit).
- The amount of the dispute exceeds the total amount of the bill against which the dispute applies. For example, if the bill due is $5, you cannot open a credit dispute for $6. Similarly, if the bill due -$5, you cannot open a debit dispute for $6.

Bill disputes can apply to the entire bill or to specific bill items. When PCM_OP_AR_BILL_DISPUTE receives an input flist, it creates a single, umbrella dispute item for all bill items covered by the dispute, as follows:

- If the input flist specifies individual bill items, the dispute item is associated with each of these bill items. Further, the dispute amount for each bill item is equivalent to the entire due for that item unless otherwise specified in the input flist.
- If the flist does not specify any bill items, all bill items are eligible for dispute. PCM_OP_AR_BILL_DISPUTE opens disputes for as many bill items as it can before it consumes the dispute amount. In this case, the dispute item will cover only those bill items that the opcode was able to fully or partially dispute before using up the dispute amount. If

the input flist does not supply a dispute amount, BRM uses amounts equivalent to the entire due for each dispute item.

In either case, PCM_OP_AR_BILL_DISPUTE passes the POID of the dispute item and the appropriate dispute amount to PCM_OP_AR_ITEM_DISPUTE, which opens the dispute for each of the associated bill items and creates the dispute events. An **/event/billing/dispute/item** object is created for each disputed item in the bill, recording the balance impact. The bill dispute event balance impact is applied to the default balance group of the bill unit associated with the bill. For details on how PCM_OP_AR_ITEM_DISPUTE works, see "Disputing Items".

Before passing the dispute item and amount to PCM_OP_AR_ITEM_DISPUTE, PCM_OP_AR_BILL_DISPUTE determines whether the dispute has tax implications. If so, it checks the original **/item** object to determine the taxable portion of the item:

- If the item object stores only an amount generated by a tax event, the opcode omits it from tax calculation.
- If the item object stores only an amount generated by a usage event, the opcode calculates taxes for the amount.
- If the item object stores an amount generated by a tax event and an amount generated by a usage event, the opcode calculates the dispute tax reversal only for the usage amount and not the tax amount.

For each item that can be disputed, PCM_OP_AR_BILL_DISPUTE checks the events that make up the item to determine whether any of the events are tax exempt. If so, it omits these events from the amount it uses when calculating the dispute tax reversal. As a result of eliminating any nontaxable components of the original item, the dispute tax is calculated proportionally to the actual taxable amount of the item.

If the dispute fails, PCM_OP_AR_BILL_DISPUTE returns the reason in the PIN_FLD_DESCR field of the PIN_FLD_RESULTS array on the output flist.

BRM creates a G/L ID for every bill dispute. For information on disputes and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Bill Disputes

Set these fields in the PCM_OP_AR_BILL_DISPUTE input flist when creating bill disputes:

- PIN_FLD_POID for the bill under dispute. You obtain this from the **/bill** object that the CSR identifies through the client application. The disputed amount is applied to the default balance group of the bill unit associated with the bill.
- PIN_FLD_POID for the item under dispute. This field is only required to dispute a particular set of bill items in the bill.
- PIN_FLD_AMOUNT for the dispute amount. The value supplied for this field should be negative when crediting and positive when debiting. This field is optional.

  You can provide the PIN_FLD_AMOUNT field for the bill itself or, if the dispute is against an array of bill items, for the individual bill items. If PIN_FLD_AMOUNT is NULL for a dispute, BRM assumes that the dispute is for the entire amount of the bill or the individual bill items, as appropriate. If you are performing a dispute that credits the bill, the absolute value of PIN_FLD_AMOUNT must be greater than 0 and less than the total bill amount and its sign should be negative.
- PIN_FLD_CURRENCY for the balance to dispute.

PCM_OP_AR_BILL_DISPUTE accepts a variety of other fields used to define tax treatment, dispute time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Bill Disputes

Use the following flags with PCM_OP_AR_BILL_DISPUTE:

- If the PIN_AR_WITHOUT_TAX flag is set, the dispute is nontaxable.
- If the PIN_AR_WITH_TAX flag is set, the dispute is taxable. Tax is reversed on an item-by-item basis.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, **fm_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The dispute is nontaxable if the **pin.conf entry** is absent.

For information on flags that indirectly affect PCM_OP_AR_BILL_DISPUTE, see "Flags for Item Disputes".

Settling Disputed Bills

To settle a dispute at the bill level, BRM uses PCM_OP_AR_BILL_SETTLEMENT. This opcode debits or credits a currency balance for the specified **/bill** object. You can create a bill settlement with or without tax. This opcode is called by BRM client applications to settle the disputed items associated with the specified bill.

For information on how PCM_OP_AR_BILL_SETTLEMENT performs settlement taxation, see "Including Taxes in the Dispute or Settlement".

> **Note:**
>
> PCM_OP_AR_BILL_SETTLEMENT does not check to determine whether the item you are settling is billed yet. However, using this opcode to settle pending items is not recommended. If you try to settle pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any disputes settled against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_BILL_SETTLEMENT *does not* create settlements in these cases:

- The specified bill is not an A/R bill (for example, a bill from a nonpaying bill unit).
- The amount of the settlement exceeds the amount of the associated dispute.

Bill settlements can apply to the entire bill or to specific bill items. When PCM_OP_AR_BILL_SETTLEMENT receives an input flist, it creates a single, umbrella settlement item for all bill items covered by the settlement, as follows:

- If the input flist specifies individual bill items, the settlement item is associated with each of these bill items. The settlement amount for each bill item must be specified in the input flist.
- If the flist does not specify any bill items, all disputed items in the bill are eligible for settlement. PCM_OP_AR_BILL_SETTLEMENT opens settlements for as many disputed items as it can before it consumes the settlement amount. After it uses up the settlement amount, it continues to open settlements for the remaining disputed items, but with a settlement amount of 0.

For each bill item it must settle, PCM_OP_AR_BILL_SETTLEMENT passes the POID of the settlement item and the appropriate settlement amount to PCM_OP_AR_ITEM_SETTLEMENT, which creates a settlement event for each bill item it receives. An **/event/billing/settlement/item** object is created for each settled item in the bill, recording the balance impact. For details on how PCM_OP_AR_ITEM_SETTLEMENT works, see "Settling Disputed Items".

If the settlement fails, PCM_OP_AR_BILL_SETTLEMENT returns the reason in the PIN_FLD_DESCR field of the PIN_FLD_RESULTS array on the output flist.

BRM creates a G/L ID for every bill settlement. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist

Set these fields in the PCM_OP_AR_BILL_SETTLEMENT input flist when creating bill settlements:

- PIN_FLD_POID for the bill under dispute or settlement. You obtain this from the **/bill** object that the CSR identifies through the BRM client application. The settled amount is applied to the default balance group of the bill unit associated with the bill.
- PIN_FLD_POID for the item to be settled. This field is only required to settle a particular set of bill items in the bill.
- PIN_FLD_AMOUNT for the settlement amount. The value supplied for this field should be negative when crediting and positive when debiting.

  You can provide the PIN_FLD_AMOUNT field for the bill itself or, if the settlement is against an array of bill items, for the individual bill items. If you supply a value for PIN_FLD_AMOUNT, it must be less than the dispute amount.

PCM_OP_AR_BILL_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Bill Settlements

Use this flag with PCM_OP_AR_BILL_SETTLEMENT:

- If the PIN_AR_WITHOUT_TAX flag is set, the settlement is nontaxable. This is the default.
- If the PIN_AR_WITH_TAX flag is set, the settlement is taxable. Tax is reversed on an item-by-item basis.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, **fm_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The settlement is nontaxable if the **pin.conf** entry is absent.

For information on flags that indirectly affect PCM_OP_AR_BILL_SETTLEMENT, see "Flags for Item Settlements".

Disputing Items

To open an item dispute, BRM uses PCM_OP_AR_ITEM_DISPUTE. This opcode debits or credits a currency balance for a particular bill item. This opcode is also called by PCM_OP_AR_BILL_DISPUTE to dispute items in a bill.

> **Note:**
>
> PCM_OP_AR_ITEM_DISPUTE does not check to determine whether the item you are disputing is billed yet. However, using this opcode to dispute pending items is not recommended. If you try to dispute pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any disputes against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_ITEM_DISPUTE does the following:

1. Reads the original **/item** object and prepares the changes to reflect the balance impact of the dispute.

   For credit disputes, these changes reduce the Due amount (PIN_FLD_DUE) for the item. For debit disputes, the opposite occurs.

2. Calls PCM_OP_BILL_POL_VALID_DISPUTE to validate the changes.

   You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT. For information on customizing the opcode, see "Customizing Item Disputes".

3. Checks to see if the PIN_FLD_ITEM_OBJ field is present in the input flist.

   If so, PCM_OP_AR_ITEM_DISPUTE checks the contents of the field to determine:

   - Whether the field contains a POID, indicating that the opcode is being called by PCM_OP_AR_BILL_DISPUTE to complete a bill dispute.

     This POID identifies the **/item/dispute** object that PCM_OP_AR_BILL_DISPUTE created to store information on the bill dispute. This is the **/item/dispute** object that PCM_OP_AR_ITEM_DISPUTE will associate with each of the dispute events it creates for the bill adjustment.

- Whether the field is NULL, indicating that PCM_OP_AR_ITEM_DISPUTE is being called directly from the client application to perform an item dispute. In this case, PCM_OP_AR_ITEM_DISPUTE will create its own **/item/dispute** object. This object is exclusive to the individual item being disputed.

4. If the dispute has tax implications and the opcode is not being called as part of a bill dispute, checks the original **/item** object to determine the taxable portion of the item:
   - If the item object stores only an amount generated by a tax event, the opcode omits it from tax calculation.
   - If the item object stores only an amount generated by a usage event, the opcode calculates taxes for the amount.
   - If the item object stores an amount generated by a tax event and an amount generated by a usage event, the opcode calculates the dispute tax reversal only for the usage amount and not the tax amount.

   For each item that it can dispute, PCM_OP_AR_ITEM_DISPUTE checks the events that make up the item to determine whether any of the events are tax exempt. If so, it omits these events from the amount it uses when calculating the dispute tax reversal.

   As a result of eliminating any nontaxable components of the original item, the dispute tax is calculated proportionally to the actual taxable amount of the item.

   > **Note:**
   >
   > If the item dispute is being performed as part of a bill dispute, PCM_OP_AR_BILL_DISPUTE performs this step before calling PCM_OP_AR_ITEM_DISPUTE.

5. Creates an **/event/billing/dispute/item** object.

   The item dispute event balance impact is applied to the default balance group of the bill unit associated with the item. If the item dispute is part of a bill dispute, the event object for each disputed item is associated with a single, umbrella **/item/dispute** object.

6. Examines the flags in the input flist to determine whether the dispute has any tax implications.

   For information on how the opcode performs dispute taxation, see "Including Taxes in the Dispute or Settlement".

7. Calls PCM_OP_BILL_TRANSFER to transfer funds to the A/R bill.
8. Writes the modified version of the original **/item** object.
9. Provides feedback on whether the dispute was successfully created.

   If the dispute creation failed, it indicates the reason for the failure.

BRM creates a G/L ID for item disputes. For information on disputes and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist

Set these fields in the PCM_OP_AR_ITEM_DISPUTE input flist when creating item disputes:

- PIN_FLD_POID for the item under dispute. The disputed amount is applied to the default balance group of the bill unit associated with the item.

- PIN_FLD_AMOUNT for the dispute amount. The value for this field should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the balance to dispute.

PCM_OP_AR_ITEM_DISPUTE accepts a variety of other fields used to define tax treatment, dispute time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Item Disputes

Use the following flags with PCM_OP_AR_ITEM_DISPUTE:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_ITEM_DISPUTE returns not only the POID, but all the fields in the **/event/billing/dispute/item** object as well.
- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_ITEM_DISPUTE does not change any fields in the database and does not create an **/event/billing/dispute/item** object. However, it does provide a dispute calculation to the caller by returning the fields that would have been used to create the event object and dispute item.

> **Note:**
>
> If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_ITEM_DISPUTE behaves in a standard manner, creating an **/event/billing/dispute/item** object to record details of the operation.

- If the PIN_AR_WITHOUT_TAX flag is set, the item dispute is nontaxable.
- If the PIN_AR_WITH_TAX flag is set, the item dispute is taxable. Tax is reversed on an item-by-item basis.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, **fm_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The dispute is nontaxable if the **pin.conf** entry is absent.

Customizing Item Disputes

When you initiate a bill or item dispute, PCM_OP_AR_ITEM_DISPUTE calls PCM_OP_BILL_POL_VALID_DISPUTE to validate all proposed changes to the fields in the original **/item** object. PCM_OP_BILL_POL_VALID_DISPUTE checks the validity of field values before processing. Field values either pass or fail. If one field fails, the entire operation fails.

You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail. PCM_OP_BILL_POL_VALID_SETTLEMENT is called by PCM_OP_AR_ITEM_SETTLEMENT.

Settling Disputed Items

To settle a dispute at the item level, BRM uses PCM_OP_AR_ITEM_SETTLEMENT. This opcode debits or credits the currency balance of a particular item to settle a dispute against that item. PCM_OP_AR_ITEM_SETTLEMENT is also called by PCM_OP_AR_BILL_SETTLEMENT to settle the dispute items associated with a bill.

> **Note:**
>
> PCM_OP_AR_ITEM_SETTLEMENT does not check to determine whether the item you are settling is billed yet. However, using this opcode to settle pending items is not recommended. If you try to settle pending items, you may introduce problems if you move the balance group to a new bill unit and there may be issues with G/L reporting.

> **Note:**
>
> You cannot move a balance group to a new bill unit if there are any disputes settled against pending bill items. Otherwise, there may be issues with G/L reporting. You can move the balance group when the items are billed.

PCM_OP_AR_ITEM_SETTLEMENT does the following:

1. Reads the original **/item** object and prepares the changes to reflect the balance impact based on the settlement amount.

   Settlement amounts have components:

   - **Granted amount**: The portion of the disputed amount granted by the settlement. This is the amount in the PIN_FLD_DISPUTED field of the PCM_OP_AR_ITEM_SETTLEMENT input flist.
   - **Denied amount**: The portion of the disputed amount that was denied by the settlement. This is the original dispute amount minus the granted amount. Settlements always work with the denied amount.

   For credit settlements, PCM_OP_AR_ITEM_SETTLEMENT makes the following changes to the balance impact:

   - Increases the Due amount (PIN_FLD_DUE) for the item by adding the amount denied during the settlement, if any.
   - For item disputes, it impacts the Adjusted amount (PIN_FLD_ADJUSTED) of the bill item by adding the settled amount and zeroing out the disputed amount (PIN_FLD_DISPUTED). Any dispute amount due to event disputes is not touched.

   For debit settlements, the opposite occurs.

2. Calls PCM_OP_BILL_POL_ITEM_VALID_SETTLEMENT to validate the changes.

   You can customize the validation criteria for PCM_OP_BILL_POL_ITEM_VALID_SETTLEMENT.

   For information on customizing the opcode, see "Customizing Item Settlements".

3. Checks to see if the PIN_FLD_ITEM_OBJ field is present in the input flist.

If so, PCM_OP_AR_ITEM_SETTLEMENT checks the contents of the field to determine:

- Whether the field contains a POID, indicating that PCM_OP_AR_ITEM_SETTLEMENT is being called by PCM_OP_AR_BILL_SETTLEMENT to complete a bill settlement.

  This POID identifies the **/item/settlement** object that PCM_OP_AR_BILL_SETTLEMENT created to store information on the bill settlement. This is the **/item/settlement** object that PCM_OP_AR_ITEM_SETTLEMENT associates with each of the settlement events it creates for the bill settlement.

- Whether the field is NULL, indicating that PCM_OP_AR_ITEM_SETTLEMENT is being called directly from the client application to perform an item settlement. In this case, PCM_OP_AR_ITEM_SETTLEMENT creates its own **/item/settlement** object. This object is exclusive to the individual item being settled.

4. Creates an **/event/billing/settlement/item** object.

   If the item settlement is part of a bill settlement, the event object for each settled item is associated with a single, umbrella **/item/settlement** object.

5. Examines the flags in the input flist to determine whether the dispute has any tax implications.

   For information on how PCM_OP_AR_ITEM_SETTLEMENT performs settlement taxation, see "Including Taxes in the Dispute or Settlement".

6. Calls PCM_OP_BILL_TRANSFER to transfer funds to the A/R bill.

   Before transferring the funds, PCM_OP_AR_ITEM_SETTLEMENT filters disputed events that were created by event disputes and subtracts the amount associated with these events from the PIN_FLD_DISPUTED of the disputed item. What remains is the amount associated solely with the item dispute, with no contribution from event disputes.

7. If the Due amount after settlement is **0** and there are no other open disputes, PCM_OP_AR_ITEM_SETTLEMENT sets the PIN_ITEM_STATUS field to CLOSED.

8. Writes the modified version of the original **/item** object.

9. Provides feedback on whether the settlement was successfully created.

   If the settlement creation failed, the opcode indicates the reason for the failure.

BRM creates a G/L ID for item settlements. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Item Settlements

Set these fields in the PCM_OP_AR_ITEM_SETTLEMENT input flist when creating item settlements:

- PIN_FLD_POID for the item to be settled. The settled amount is applied to the default balance group of the bill unit associated with the item.
- PIN_FLD_AMOUNT for the settlement amount. The value for this field should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the balance to settle.

PCM_OP_AR_ITEM_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Item Settlements

Use these flags with PCM_OP_AR_ITEM_SETTLEMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_ITEM_SETTLEMENT returns not only the POID, but all the fields in the **/event/billing/settlement/item** object as well.
- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_ITEM_SETTLEMENT does not change any fields in the database and does not create an **/event/billing/settlement/item** object. However, it does provide a settlement calculation to the caller by returning the fields that would have been used to create the event object and settlement item.

> **Note:**
>
> If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_ITEM_SETTLEMENT behaves in a standard manner, creating an **/event/billing/settlement/item** object to record details of the operation.

- If the PIN_AR_WITHOUT_TAX flag is set, the item settlement is nontaxable.
- If the PIN_AR_WITH_TAX flag is set, the item settlement is taxable. Tax is reversed on an item-by-item basis.

> **Note:**
>
> If the tax flag is not set, BRM refers to the pin.conf entry, **fm_ar tax_reversal_with_tax**, to determine the default tax reversal behavior. The settlement is nontaxable if the **pin.conf** entry is absent.

Customizing Item Settlements

When you initiate a bill or item settlement, PCM_OP_AR_ITEM_SETTLEMENT calls PCM_OP_BILL_POL_VALID_DISPUTE to validate all proposed changes to the fields in the original **/item** object. This opcode checks the validity of field values before processing. Field values either pass or fail. If one field fails, the entire operation fails.

You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

Disputing Events

To open a dispute at the event level, BRM uses PCM_OP_AR_EVENT_DISPUTE.

This opcode receives a batch of events on the input flist, and returns the disputed **/event/billing/dispute/event** data on the output flist. There can be multiple events in a given batch, and the events to be disputed do not need to belong to the account specified in the PIN_FLD_POID for PCM_OP_AR_EVENT_DISPUTE. The input flist can optionally include a reason code domain and descriptive string that indicates why the dispute is being made.

> **Caution:**
>
> PCM_OP_AR_EVENT_DISPUTE does not verify whether there are other disputes against the events it processes. Multiple disputes for the same event can cause conflicts in BRM, so make sure your client application prevents the CSR from logging more than one dispute for an event.

When it receives a batch of events, PCM_OP_AR_EVENT_DISPUTE performs these tasks:

1. It reads the batch, identifying all events to be disputed.
2. It examines the flags in the input flist to determine whether the dispute has any tax implications.

   For information on how the opcode performs dispute taxation, see "Including Taxes in the Dispute or Settlement".
3. It creates a single dispute item, **/item/dispute**, for all the disputed events in the batch.
4. For each event to be disputed, it creates a new **/event/billing/dispute/event** object containing the dispute amount.

   The PIN_FLD_ITEM_OBJ field of **/event/billing/dispute/event** points to the dispute item created in step 3.
5. After the **/item/dispute** and **/event/billing/dispute/event** objects are created, it calls PCM_OP_BILL_ITEM_TRANSFER to transfer the dispute amount from the dispute item to the original item that owns the disputed event.

   > **Note:**
   >
   > If the opcode transfer targets a closed item, that item is reopened automatically.
6. It creates a notification event (**/event/billing/dispute/notify**).

   If event notification is enabled in your system, this triggers the reservation of the disputed balances until the dispute is settled.
7. When it finishes, it provides feedback on whether the dispute was successfully created or, if dispute creation failed, the reason for the failure.

BRM creates a G/L ID for event disputes. For information on disputes and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Event Disputes

Set these fields in the PCM_OP_AR_EVENT_DISPUTE input flist when creating event disputes:

- PIN_FLD_POID for each of the events to be disputed. These fields form a PIN_FLD_EVENTS array. The disputed amount is applied to the default balance group of the bill unit associated with the **/item** object that the event belongs to.
- PIN_FLD_AMOUNT for the dispute amount. The value supplied for this field should be positive for adjustments that credit the event and negative for adjustments that debit the event.

You can specify the dispute amount through either PIN_FLD_AMOUNT or PIN_FLD_PERCENT. If you omit both of these fields, BRM disputes the entire amount for each event.

- PIN_FLD_RESOURCE_ID for the disputed balance.

PCM_OP_AR_EVENT_DISPUTE accepts a variety of other fields used to define tax treatment, dispute time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Event Disputes

Use the following flags with PCM_OP_AR_EVENT_DISPUTE:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_EVENT_DISPUTE returns not only the POID, but all the fields in the **/event/billing/dispute/event** object as well.
- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_EVENT_DISPUTE does not change any fields in the database and does not create an **/event/billing/dispute/event** object. However, it does provide a dispute calculation to the caller by returning the fields that would have been used to create the event object and dispute item.

> **Note:**
>
> If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_EVENT_DISPUTE behaves in a standard manner, creating an **/event/billing/dispute/event** object to record details of the operation.

- If the PIN_ADJ_NO_TAX flag is set, the event dispute is nontaxable. If this flag is not set, the dispute is taxable.
- If the PIN_ADJ_TAX_ONLY flag is set, only the tax on the original event is disputed; not the event itself.

> **Note:**
>
> Do not set this flag for disputes against unbilled events that use deferred taxation.

- If the PIN_EVENT_ADJUST_BATCH is set, the dispute amount is distributed among the events in the batch on a first-come-first-serve basis until the dispute amount is consumed.
- If the PIN_EVENT_ADJUST_EVENT is set, the dispute amount is applied individually to each of the events in the batch.

Settling Disputed Events

To settle a dispute at the event level, BRM uses PCM_OP_AR_EVENT_SETTLEMENT. This opcode debits or credits the balances of an event to settle a dispute against that event.

> **Note:**
>
> You cannot settle an event more than once. If PCM_OP_AR_EVENT_SETTLEMENT settles an event that has more than one open dispute, it settles all the disputes for that event.

PCM_OP_AR_EVENT_SETTLEMENT receives dispute item POID in the input flists and retrieves the associated dispute events. It returns the settled **/event/billing/settlement/event** data on the output flist. The input flist can optionally include a reason code domain and descriptive string that indicates why the settlement is being made.

When it receives a batch of events, PCM_OP_AR_EVENT_SETTLEMENT performs these tasks:

1. It verifies that, for backdated settlements, the settlement date falls after the dispute date.
2. It examines the flags in the input flist to determine whether the settlement has any tax implications.

   For information on how the opcode performs dispute taxation, see "Including Taxes in the Dispute or Settlement".
3. It searches for all dispute events whose PIN_FLD_ITEM_OBJ field points to the dispute item and creates a single settlement item, **/item/settlement**, for all these events.
4. For each dispute event to be settled, it creates a new **/event/billing/settlement/event** object containing the denied amount.

   The PIN_FLD_SESSION_OBJ field of **/event/billing/settlement/event** points to the corresponding dispute event.
5. After the **/item/settlement** and **/event/billing/settlement/event** objects are created, it calls PCM_OP_BILL_ITEM_TRANSFER to transfer the denied amount from the settlement item to the original item against which the settlement was filed and recalculate the Due of the original item accordingly.

   > **Note:**
   >
   > If PCM_OP_AR_EVENT_SETTLEMENT transfer targets a closed item, that item is reopened automatically.

   To prevent misallocation of balances during settlement, PCM_OP_AR_EVENT_SETTLEMENT calculates the combined dispute amount for all the dispute events with respect to the original disputed item. This amount is associated solely with the event dispute being settled. It has no contribution from any item disputes or any event disputes other than the one being settled. It is this amount that PCM_OP_BILL_ITEM_TRANSFER works with when transferring the item.
6. It creates a notification event (**/event/billing/settlement/notify**).

   If event notification is enabled in your system, this triggers the release of disputed balances.
7. When it finishes, PCM_OP_AR_EVENT_SETTLEMENT provides feedback on whether the settlement was successfully created or, if settlement creation failed, the reason for the failure.

BRM creates a G/L ID for event settlements. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Event Settlements

Set these fields in the PCM_OP_AR_EVENT_SETTLEMENT input flist when creating event settlements:

- PIN_FLD_ITEM_OBJ for the dispute item.
- PIN_FLD_AMOUNT for the settlement amount. The value supplied for this field should be positive for settlements that credit the event and negative for settlements that debit the event.

  You can specify the settlement amount through the PIN_FLD_AMOUNT field or omit the field altogether. If you omit this field, BRM settles the entire amount for each event.
- PIN_FLD_RESOURCE_ID for the balance to settle.

PCM_OP_AR_EVENT_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so on.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".

Flags for Event Settlements

Use the following flags with PCM_OP_AR_EVENT_SETTLEMENT:

- If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_AR_EVENT_SETTLEMENT returns not only the POID, but all the fields in the **/event/billing/settlement/event** object as well.
- If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_AR_EVENT_SETTLEMENT does not change any fields in the database and does not create an **/event/billing/settlement/event** object. However, it does provide a settlement calculation to the caller by returning the fields that would have been used to create the event object and settlement item.

  > **Note:**
  >
  > If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_AR_EVENT_SETTLEMENT behaves in a standard manner, creating an **/event/billing/settlement/event** object to record details of the operation.
- If the PIN_ADJ_NO_TAX flag is set, the event settlement is nontaxable. If this flag is not set, the settlement is taxable.
- If the PIN_ADJ_TAX_ONLY flag is set, only the tax on the original event is settled; not the event itself.

  > **Note:**
  >
  > Do not set this flag for settlements against unbilled events that use deferred taxation.

- If the PIN_EVENT_ADJUST_BATCH is set, the settlement amount is distributed among the events in the batch on a first-come-first-serve basis until the settlement amount is consumed.
- If the PIN_EVENT_ADJUST_EVENT is set, the settlement amount is applied individually to each of the events in the batch.

Configuring the Tax Treatment for Adjustments, Disputes, and Settlements

To ensure that you can control whether BRM applies adjustment, dispute, and settlement tax reversals, your custom CRM implementation should set the following flags in the input flists for all account, bill, and item level adjustment, dispute, and settlement opcodes:

- **PIN_AR_WITH_TAX**: Set this flag for all adjustments, disputes, and settlements that involve a tax reversal.
- **PIN_AR_WITHOUT_TAX**: Set this flag for adjustments, disputes, and settlements that do not require a tax reversal.

If you do not pass in one of the flags, BRM uses the **fm_ar tax_reversal_with_tax** entry in the CM configuration file to determine the default behavior. This entry takes one of the tax flags (PIN_AR_WITH_TAX or PIN_AR_WITHOUT_TAX). If you do not pass in one of the flags and this entry is not present, BRM does not perform a tax reversal.

> **Note:**
>
> Event adjustments, disputes, and settlements use different flags for tax reversal. For more information on tax flags for event opcodes, see Flags for Event Adjustments, "Flags for Event Disputes", and "Flags for Event Settlements".

Including Taxes in the Dispute or Settlement

You specify an optional tax treatment for disputes and settlements through flags in the dispute and settlement opcode input flists. For information, see:

- Flags for Bill Disputes
- Flags for Bill Settlements
- Flags for Item Disputes
- Flags for Item Settlements
- Flags for Event Disputes
- Flags for Event Settlements

Tax Processing for Disputes

The dispute opcodes calculate taxes when the dispute is created. Before calculating taxes, they validate that the item due is not zero. If it is zero, the dispute is not allowed.

If the input flist flags the dispute or settlement as taxable, the dispute opcode either includes or excludes the tax:

- If it is tax now, the opcode uses information from the disputed event to calculate the tax and apply it immediately.

- If it is tax deferred, taxes are contained in the tax item. In such cases, the opcode does the following to handle the taxes:
  - Calls PCM_OP_BILL_TAX_EVENT to calculate the tax amount.
  - Compares the calculated tax with due amount of the tax item to determine the amount to reverse.
  - Creates the **/event/billing/dispute/tax_event** event to contain the tax amount.
  - Transfers the calculated tax amount to the original tax item.

> **Note:**
>
> If the dispute is created in the same billing cycle as the disputed item, the original item will not have been taxed yet. In this case, there is no need to create the **/event/billing/dispute/tax_event** or **/event/billing/settlement/tax_event** event because the dispute or settlement precedes initial tax calculation for the original item.

Tax Processing for Settlements

The settlement opcodes calculates taxes when the settlement is created. Before calculating taxes, they validate that the settlement amount is less than or equal to the disputed amount. If the settlement amount is greater, the settlement is not allowed.

If the original bill item uses taxation, the tax is considered tax inclusive or tax exclusive, depending on the amount of the dispute:

- If the disputed amount is partially granted, the settlement amount is considered to be tax-exclusive. The amount due on the bill item is increased by the net denied amount and the amount due on the cycle tax item is increased by the denied tax amount.
- If the disputed amount is completely granted (there is no denied amount), the settlement amount is considered to be tax-inclusive. The amount due on the bill item does change for taxes.

> **Note:**
>
> If the disputed amount is completely denied, the amount due on the bill item is increased by the net amount and the amount due on the tax item is increased by the tax amount.

If the input flist flags the settlement as taxable, the settlement opcode determines whether the tax method is tax now or tax deferred.

- If the tax method is tax now, the settlement opcodes use information from the event associated with the original bill item to calculate taxes and apply the amount immediately with the settlement amount.
- If the tax method is tax deferred, taxes are contained in the tax item associated with the dispute. In such cases, the opcode does the following to handle the taxes:
  - Calls PCM_OP_BILL_TAX_EVENT to calculate the net and tax amount from the total settlement amount. If a denied amount exists, it then calculates the net and tax amount from the total denied amount.

  - Creates the **/event/billing/settlement/tax_event** event to contain the denied tax amount.
  - Transfers the denied net amount to the original item and the denied tax amount to the tax item.

Including Reason Codes in the Dispute or Settlement

Reason codes explain why a dispute is being opened or settled. In typical implementations, the BRM client application provides a selectable set of reasons appropriate to the event type. You choose a reason that matches the situation or, in some cases, compose a new reason. The client application populates the input flist for the opcode with this information, and BRM records the reason in the dispute or settlement event object (for example, **/event/billing/dispute/item** or **/event/billing/settlement/event**). The dispute and settlement opcode flist fields that provide reason code information are PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID. These fields are optional.

For general information about reason codes, see String Manipulation Functions in *BRM Developer's Reference*.

Assigning G/L IDs for a Dispute or Settlement

The G/L ID for a dispute or settlement event is assigned like other pre-rated events in the BRM system. PCM_OP_ACT_POL_SPEC_GLID assigns the G/L ID for the adjustment.

Writing Off Debts and Reversing Write-Offs

Write-offs remove A/R amounts that your company determines will never be paid. Write-off reversals allocate payments to accounts, bills, bill units, and items that have been previously written off. Write-offs are a standard feature of your BRM system.

About Initiating Write-Offs

BRM uses the following opcodes to perform write-offs:

- PCM_OP_AR_ACCOUNT_WRITEOFF. Writes off all A/R bill units for an account and each bill unit's nonpaying children. This opcode performs a write-off when there are open items with due amounts and your company has determined that these items will never be paid by the customer. To have your client application initiate an A/R account write-off, it should call this opcode and the input flist should specify the account POID.
- PCM_OP_AR_BILLINFO_WRITEOFF. Writes off an A/R bill units for an account. To have your client application perform an A/R bill unit write-off, it should call this opcode with the bill unit specified in the input flist.
- PCM_OP_AR_BILL_WRITEOFF. Writes off a bill. To have your client application perform a bill write-off, it should call this opcode with the bill specified in the input flist.
- PCM_OP_AR_ITEM_WRITEOFF. Writes off a bill item. To have your client application perform an item write-off, it should call this opcode with one or more items specified in the input flist.

> **Note:**
>
> – Before you write off an account or bill unit, ensure that all items have been billed and that there are no pending items for the account or bill unit.
>
> – To ensure that you can reverse a write-off on an account, the account status must be inactive before you write off the account.

Performing Write Offs

Though you can initiate a write-off by using opcodes for the account, bill unit, bill, or item, most write-offs are performed by using PCM_OP_AR_ITEM_WRITEOFF. This opcode is called in one of the following ways:

- By using PCM_OP_AR_BILLINFO_WRITEOFF and PCM_OP_AR_BILL_WRITEOFF when you perform an account, bill unit, or bill write-off

  > **Note:**
  >
  > PCM_OP_AR_ACCOUNT_WRITEOFF, PCM_OP_AR_BILLINFO_WRITEOFF, and PCM_OP_AR_BILL_WRITEOFF perform several initial steps before calling PCM_OP_AR_ITEM_WRITEOFF. For more information, see "About Account Write-Offs", "About Bill Unit Write-Offs", and "About Bill Write-Offs".

- Independently when you perform an item write-off

  > **Note:**
  >
  > PCM_OP_AR_ITEM_WRITEOFF performs write-offs on pending items with nonzero balances only.

PCM_OP_AR_ITEM_WRITEOFF does the following:

1. Prepares the charges for the **/item** object (for example, **/item/misc**).
2. Calls PCM_OP_BILL_POL_ITEM_VALID_WRITEOFF to validate the changes.

   For information on how to customize write-off validation, see "Customizing Write-Off Validation".
3. Creates the new write-off item (**/item/writeoff**).

   For an account or bill write-off, the item total is the sum of all item charges in the account or bill.
4. Creates the **/event/billing/writeoff/item** object.

   The balance impact of the write-off event is the sum of all amounts due for all open items in the bill unit.

   - If an account is written off, an **/event/billing/writeoff/account** object is created for the account and an **/event/billing/writeoff/billinfo** object is created for each bill unit.
   - If a bill is written off, an **/event/billing/writeoff/bill** object is created.

5. If you have specified to store the tax amount of the write-off in a separate event, PCM_OP_AR_ITEM_WRITEOFF creates one of the following event objects:
   - **/event/billing/writeoff/tax_billinfo**
   - **/event/billing/writeoff/tax_bill**
   - **/event/billing/writeoff/tax_item**

   For information on tax treatments for write-offs, see "About Taxes for Write-Offs".
6. Calls PCM_OP_BILL_TRANSFER to transfer funds to the A/R bill.

   Multiple transfer events (**/event/billing/item/transfer**) are created to move the credit from the write-off item to all the open items for the bill unit.
7. If the item has no due or disputed amount, PCM_OP_AR_ITEM_WRITEOFF closes the item.
8. Writes the modified **/item** object.

When this process is complete, all the open items, including the write-off item, are closed. The G/L ID for the account write-off is assigned by calling PCM_OP_ACT_POL_SPEC_GLID, which assigns the G/L ID for the write-off event (**/event/billing/item/writeoff**).

If write-off reversal is enabled, PCM_OP_AR_ITEM_WRITEOFF stores the write-off item in the write-off profile object of the account.

> **Note:**
>
> - Before you write off an account or bill unit, ensure that all items have been billed and that no pending or unbilled items for the account or bill unit remain.
> - To ensure that you can reverse a write-off on an account, the account status must be inactive before you write off the account.

About Account Write-Offs

To write off an account, PCM_OP_AR_ACCOUNT_WRITEOFF writes off *all* the bill units associated with the account.

This opcode performs the following error checks before performing the item write-offs for the account:

- Ensures the due amount is not zero. Bill units with zero due are fully paid and do not need to be written off.
- Ensures that there are no partially allocated items (for example, a refund must be allocated before the write-off is performed).

PCM_OP_AR_ACCOUNT_WRITEOFF then calls PCM_OP_AR_BILLINFO_WRITEOFF to write off all the open items for each bill unit of the account.

The **/event/billing/writeoff/account** event is created to record the account write-off. The **/event/billing/writeoff/billinfo** event is created to record each bill unit write-off, and **/event/billing/writeoff/tax_billinfo** is created for each bill if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as PIN_BILL_WRITEOFF_TAX in the input flist.

Your custom application should call PCM_OP_AR_ACCOUNT_WRITEOFF, not PCM_OP_AR_BILLINFO_WRITEOFF, to write off *all* bill units in an account.

About Bill Unit Write-Offs

PCM_OP_AR_BILLINFO_WRITEOFF performs write-off adjustments for a specific bill unit when there are open items with due amounts and it has been determined that these items will never be paid by a customer.

To write-off *all* bill units of an account, use PCM_OP_AR_ACCOUNT_WRITEOFF.

To write off a bill unit, PCM_OP_AR_BILLINFO_WRITEOFF performs the following error checks before performing the item write-offs for the bill unit:

- Ensures the due amount is not zero. Bill units with zero due are fully paid and do not need to be written off.
- Ensures that there are no partially allocated items (for example, a refund must be allocated before the write-off is performed).

PCM_OP_AR_BILLINFO_WRITEOFF then calls PCM_OP_AR_ITEM_WRITEOFF to write off each of the bill items in the bill unit.

When a bill unit is written off, a write-off item (**/item/writeoff**) is created. The total for this item is the sum of the balance due from all the open items for the A/R bill units and its nonpaying child bill units. This is the balance impact of the write-off. The write-off decreases the account balances and the transfer event moves the credit from the write-off item to the items being written off.

The **/event/billing/writeoff/billinfo** event is created to record a bill unit write-off, and the **/event/billing/writeoff/tax_billinfo** event is created for the bill unit if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as PIN_BILL_WRITEOFF_TAX in the input flist.

About Bill Write-Offs

PCM_OP_AR_BILL_WRITEOFF performs write-off adjustments against a specified bill when there are open items with due amounts and it has been determined that these items will never be paid by a customer. The write-off decreases the account balances associated with the specified bill, and the transfer event moves the credit from the write-off item to the item being written off. To write off a bill, PCM_OP_AR_BILL_WRITEOFF ensures the due amount is not zero. Bills with a zero due are fully paid and do not need to be written off.

PCM_OP_AR_BILL_WRITEOFF then calls PCM_OP_AR_ITEM_WRITEOFF to write off each of the bill items in the bill.

When a bill is written off, a write-off item (**/item/writeoff**) is created. The write-off decreases the account balances associated with the specified bill, and the transfer event moves the credit from the write-off item to the item being written off.

The **/event/billing/writeoff/bill** event is created to record a bill write-off, and **/event/billing/writeoff/tax_bill** is created for the bill if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as 1 in the input flist.

Flags You Should Use for Write-Offs

Set flags in the PIN_FLD_FLAGS field for the PCM_OP_AR_BILL_WRITEOFF input flist when creating write-offs. You can include these flags in any of the write-off opcodes.

- If the PIN_BILL_WRITEOFF_TAX flag is set, BRM creates a separate write-off event for taxes.

About Taxes for Write-Offs

To write off tax, BRM selects all the open items. For all open items, PCM_OP_AR_ITEM_WRITEOFF selects all events that point to these items, except for the tax events. After the events are selected, it calculates the sum of the total initial net and tax amount and groups the events based on tax rate. The write-off process starts with the lowest tax rate first and continues with the next lowest rate until it reaches the total write-off amount. The total write-off amount (VAT included) is the total due amount of the bill.

The following types of events are created for a write-off with tax:

- A write-off event holding the total net amount of the write-off with a G/L ID for the net amount
- A tax write-off event holding the VAT amount of the write-off with a G/L ID for tax amount
- Transfer events to close all the remaining open items of the bill

Customizing Write-Off Validation

You use PCM_OP_BILL_POL_VALID_WRITEOFF to customize how the item is validated for a write-off. The default implementation is:

- The item must be open.
- The write-off amount must be less than or equal to the amount due.

PCM_OP_BILL_POL_VALID_WRITEOFF is called by PCM_OP_AR_ITEM_WRITEOFF. It checks the validity of field values before processing. Field values either pass or fail. If one field fails, the entire operation fails.

You can customize the validation criteria for PCM_OP_BILL_POL_VALID_WRITEOFF by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

About Initiating Write-Off Reversals

Write-off reversals are initiated when PCM_OP_PYMT_COLLECT encounters a payment for an account or bill unit that has been written off. PCM_OP_PYMT_COLLECT runs automatically as part of the payment collection process or can be called manually. PCM_OP_PYMT_COLLECT calls PCM_OP_AR_REVERSE_WRITEOFF to perform the write-off reversal.

The following opcodes perform the actual reversal:

- PCM_OP_AR_REVERSE_WRITEOFF: This opcode reverses the write-off and opens the original bill items so that the payment can be allocated.

> **Note:**
>
> Write-off reversals should only be performed automatically; you should not call PCM_OP_AR_REVERSE_WRITEOFF directly.

- PCM_OP_BILL_POL_REVERSE_PAYMENT: This opcode writes off an account that was previously written off, but which then received a payment that reversed the original write-off. For example, if a payment that reversed the write-off of an account is determined to be fraudulent, the account will again be written off.

Reversing Write-Offs

To reverse a write-off, BRM uses PCM_OP_AR_REVERSE_WRITEOFF. This opcode is called by PCM_OP_PYMT_COLLECT.

PCM_OP_AR_REVERSE_WRITEOFF reverses write-offs on all written-off bills and bill items associated with a written-off account or bill unit when a payment for the account or bill unit is received.

PCM_OP_AR_REVERSE_WRITEOFF accepts an array of written-off items to be reversed.

To perform a reversal of a written-off bill unit, specify the POID of the written-off bill unit.

> **Note:**
>
> If you want a separate event to record the tax amount, set the PIN_BILL_WRITEOFF_TAX flag in the input flist.

> **Note:**
>
> Write-off reversals can only be performed automatically; do not call PCM_OP_AR_REVERSE_WRITEOFF directly. The automatic write-off reversal feature must be enabled for this opcode to perform write-off reversals.

PCM_OP_AR_REVERSE_WRITEOFF performs the following tasks:

1. Validates that the item in the input flist is a written-off item.

   If the input flist contains no written-off items, PCM_OP_AR_REVERSE_WRITEOFF calls PCM_OP_AR_POL_REVERSE_WRITEOFF to supply a list of write-off items that require reversal if that list is not provided by any other means.
2. Opens the originally written-off items so that funds can be allocated.
3. Creates a write-off reversal item (**/item/writeoff_reversal**).
4. Creates a write-off reversal event (**/event/billing/writeoff_reversal**) that points to the write-off reversal item.
5. Updates the account's balance.
6. (Account write-off reversal only) Sets the value of the **/profile/writeoff** object's PIN_FLD_ITEM_OBJ field to NULL.

   PCM_OP_AR_REVERSE_WRITEOFF also sets the PIN_FLD_WRITEOFF_INFO flag in this object to the reversed state (**2**). If the payment is not equal to or greater than the sum of the written-off items, this flag is reset to the write-off state (**1**).

Reversing a Write-Off Reversal

When BRM receives a payment for a written-off account or bill unit, it reverses the account or bill unit write-off and then allocates the payment. If that payment must later be reversed (for example, if it fails financially), BRM must then write off the account or bill unit a second time.

This activity is performed by PCM_OP_BILL_POL_REVERSE_PAYMENT. This opcode is called by PCM_OP_BILL_REVERSE_PAYMENT, and the input flist for the opcode includes any flags passed by the standard opcode.

> **Note:**
>
> If any open unallocated items are in the account or bill unit at the time of the reversal, the second write-off on the account does not occur. You can either allocate and close the open items before performing the reversal or customize this opcode to perform the task.

PCM_OP_BILL_POL_REVERSE_PAYMENT performs the following operations if the PIN_FLD_STATUS value on the input flist is PIN_PYMT_WRITEOFF_SUCCESS and if the transaction ID of the payment to be reversed is valid and applied to a written-off amount.

1. Write off the account or bill unit again (if the original payment was an underpayment).
2. Assign a reason code for the second write-off.
3. Set the value of the PIN_FLD_FLAGS field to PIN_BILL_WRITEOFF_TAX to generate a separate tax event (if necessary), and assign a reason code for the tax event.

The PIN_FLD_INHERITED_INFO substruct in the output flist passes additional information to the calling opcode. If necessary, PCM_OP_BILL_REVERSE_PAYMENT returns any information in this substruct to the calling opcode.

Customizing Write-Off Reversals

You can customize two aspects of write-off reversals:

- Use PCM_OP_AR_POL_REVERSE_WRITEOFF to customize the rules that govern how BRM performs write-off reversals.

  See "Customizing the Rules for Performing Write-Off Reversals".

- Use PCM_OP_BILL_POL_REVERSE_PAYMENT to customize the way that BRM processes payments that reverse account write-offs but that are, in turn, reversed. Payments of this sort result in a second write-off of the account.

  See "Customizing Reversal of Payments Allocated to Written-Off Accounts".

Customizing the Rules for Performing Write-Off Reversals

You can customize PCM_OP_AR_POL_REVERSE_WRITEOFF to implement additional rules for performing write-off reversals and allocating funds to written-off accounts and bill units. For example, you can perform the write-off reversal only for accounts that were written off during a specified time period or for a minimum amount.

PCM_OP_AR_POL_REVERSE_WRITEOFF is called by PCM_OP_AR_REVERSE_WRITEOFF during the write-off reversal process.

PCM_OP_AR_POL_REVERSE_WRITEOFF supplies a list of write-off items that require reversal if that list is not provided by any other means. The PCM_OP_AR_POL_REVERSE_WRITEOFF input flist includes any flags passed by the standard opcode. PCM_OP_AR_POL_REVERSE_WRITEOFF retrieves the write-off reversal items from the **/profile/writeoff** object.

> **Note:**
>
> If any open unallocated items are in the account or bill unit at the time of the reversal, the second write-off on the account does not occur. You can either allocate and close the open items before performing the reversal, or customize this opcode to perform the task.

About Automatic Write-Off Reversals During Payment Collection

You can configure BRM to reverse a write-off on an account or bill unit automatically when a payment is received for a written-off account or bill unit. BRM verifies that the write-off flag and write-off item are set in the account's **/profile/writeoff** object. If so, the following occurs:

1. The write-off flag in the account's **/profile/writeoff** object is set to **2**, which is the reversed state.
2. All written-off bills and bill items are opened so the funds can be allocated.
3. The **/event/billing/writeoff/billinfo** balance amount is transferred to the original bill or bill item and then set to **0**.
4. The paid bills and items are closed.
5. The write-off flag in the account's **/profile/writeoff** object is reset to **1**, which is the written-off state.

> **Note:**
>
> To ensure that you can reverse a write-off on an account, the account status must be **inactive** before you write off the account.

For information about using the PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode to customize write-off reversal rules, see "Customizing the Rules for Performing Write-Off Reversals".

Customizing Reversal of Payments Allocated to Written-Off Accounts

When a payment that was applied to a written-off account or bill unit is reversed, the previously paid balance is returned to the account balance and then written off again. This enables the original written-off amount to be reestablished as unrecoverable debt.

Table 4-5 shows an example where multiple payments are received for an account with a $100 write-off amount.

Payment 1 is $40 (underpayment), so the $100 write-off amount is reversed and $60 is written off.

Payment 2 is $100 (overpayment); therefore, the $60 write-off amount is reversed, and $40 excess (credit) amount is left unallocated in the account.

Table 4-5 Reversing Payment Applied to a Written-Off Account

Transaction	Due Amount	Unrecovered Debt	Recovered Debt	Bank Deposit
Account balance	100	NA	NA	NA
Initial write-off	-100	100	NA	NA
Write-off reversal	100	NA	-100	NA
Payment	-40	NA	NA	40
Re–write-off	-60	NA	60	NA
Write-off reversal	60	NA	-60	NA
Payment Reversal	40	NA	NA	-40
Re–write-off	-100	NA	100	NA
Net Result	**0**	**100**	**0**	**0**

> **Note:**
>
> If a subsequent payment is received during this operation, or if an A/R action is performed, the account may not be restored to its previous state. This occurs if any open unallocated items are in the account at the time of the reversal.

If Payment 1 is reversed, the write-off is *not* reversed because there is no longer a written-off amount on the account; instead, a credit in the account balance remains. To avoid this scenario, customize the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to handle the allocation and write-off. Or allocate the credit balance to any open bills or bill items, and then manually write off the account.

PCM_OP_BILL_POL_REVERSE_PAYMENT performs optional processing on payments that were applied to written-off accounts, and that must be reversed. For example, PCM_OP_BILL_POL_REVERSE_PAYMENT allocates balances on open bills and bill items before performing write-off reversals.

If any open unallocated items are in the account at the time of the reversal, the re-writeoff on the account does not occur. You can either allocate and close the open items before performing the reversal, or customize PCM_OP_BILL_POL_REVERSE_PAYMENT to perform the task.

PCM_OP_BILL_POL_REVERSE_PAYMENT is called by PCM_OP_BILL_REVERSE_PAYMENT.

You can implement special processing for reversing payments that were applied to written-off accounts. To implement this processing, you customize PCM_OP_BILL_POL_REVERSE_PAYMENT to perform the following functions:

- Assign custom reason codes.
- Generate a tax event based on custom business policies.
- *Not* reverse write-offs when a payment is applied.

- Allocate any open items in the account before performing the reversal, before the reversal can occur. This is required if multiple payments are received, and a subsequent payment results in an overpayment.

  For example, if a $100 write-off amount is present for an account and a $40 payment is received, followed by a $90 payment, a $30 credit is left unallocated on the account.

  Because write-offs are valid only for debit items, if you reverse the $40 payment, BRM creates a $40 *debit* open item and a $30 *credit* open item, and the result would be a $10 unallocated balance. This is not possible; therefore, the unallocated credit of $30 must be allocated before the re–write-off can be performed.

  To enable automatic allocation, customize PCM_OP_BILL_POL_REVERSE_PAYMENTto call PCM_OP_BILL_ITEM_TRANSFER, and allocate the open credit amount to the available debit items before performing the re–write-off.

Retrieving A/R Information

You can retrieve various kinds of balance information, such as the following:

- Current balances for items and bills
- A list of open and pending bill items
- A list of bills
- A list of adjusted items

To retrieve balance information, use the following opcodes.

- **PCM_OP_AR_GET_ACCT* opcodes**: These opcodes retrieve the current balance for open and pending bill items associated with all the bill units (**/billinfo** objects) in the account or for a specified bill unit. In the latter case, you specify the bill unit in the input flist.
- **PCM_OP_AR_GET* opcodes**: These opcodes retrieve the current balance for open and pending bill items associated with the specified bill unit. Specify the bill unit in the input flist.

> **Note:**
>
> To retrieve information for all of an account's bill units by using PCM_OP_AR_GET*, implement custom code that recursively calls PCM_OP_AR_GET* for each bill unit. This technique produces results similar to using PCM_OP_AR_GET_ACCT*, but it involves the overhead of creating a customization.

Finding a Bill

To find a bill, BRM uses PCM_OP_BILL_FIND. You can use this opcode to search for all **/bill** objects instead of using PCM_OP_SEARCH and PCM_OP_STEP_SEARCH.

> **Note:**
>
> PCM_OP_BILL_FIND does not perform authentication.

- If you specify an array of bill numbers, PCM_OP_BILL_FIND returns the POID of the **/bill** object.
- If you specify the results array, the fields in the results array are returned.
- If you do not specify the results array, PCM_OP_BILL_FIND returns the entire contents of the object.

Finding a Bill Unit

To find the bill units for an account, BRM uses PCM_OP_BAL_GET_ACCT_BILLINFO. This opcode returns the main contact information for an account and a list of the account's bill units with the default bill unit marked.

The output flist contains:

- Contact information from the **/account** object, including the account's contact name, address, and phone number.
- The POIDs, names, and payment methods of all the bill units associated with that account. If more than one bill unit is returned, the default bill unit array contains a PIN_FLD_FLAGS entry of **1**. The PIN_FLD_FLAGS entry for each of the other bill unit arrays is **0**.

Finding a Balance Group and Its Balances

To retrieve a **/balance_group** POID and, optionally, the balances it contains, BRM uses PCM_OP_BAL_GET_BALANCES. This opcode returns the **/balance_group** and **/billinfo** POIDs for an account or service. Using the PIN_FLD_BALANCES array, you can also direct this opcode to return any or all of the balances contained in the **/balance_group** object.

PCM_OP_BAL_GET_BALANCES finds the **/balance_group** POID by using the event end time and the **/service** object's PIN_FLD_TRANSFER_LIST array. For more information, see "Transferring Services between Balance Groups".

You can specify the validity period for which to retrieve sub-balances for the given balance group by setting the PIN_FLD_FLAGS field to one of the following values:

- PIN_BAL_GET_BARE_RESULTS (0x1): This flag returns all sub-balances for the current cycle that are currently valid.
- PIN_BAL_GET_ALL_BARE_RESULTS (0x4): This flag returns the sub-balances that are valid currently and in the future. To return future sub-balances, there must be at least one valid sub-balance for the current period.
- PIN_BAL_GET_BASED_ON_PERIOD (0x10): This flag returns all sub-balances whose validity period falls within or overlaps a given period, specified by the PIN_FLD_START_T and PIN_FLD_END_T fields.

  For example, if the period is June 1 through June 30, the opcode retrieves all sub-balances that were valid for any time between June 1 and June 30, whether their validity period begins before June 1 or ends after June 30.
- PIN_BAL_GET_WITHIN_PERIOD (0x16): This value returns only the sub-balances whose start and end times fall entirely within the given time period specified by the PIN_FLD_START_T and PIN_FLD_END_T fields.

  For example, if the period is June 1 through June 30, the opcode retrieves only sub-balances that start on or after June 1 and end on or before June 30.

If no balance is available for the specified period, PCM_OP_BAL_GET_BALANCES returns 0.

To return the balances that were granted by a specific charge offer or discount offer, specify the POID of the granting charge offer or discount offer in the PIN_FLD_GRANTOR_OBJ field.

PCM_OP_BAL_GET_BALANCES returns balances that are configured to start when first impacted (on first usage) if their validity periods have not yet been set. When a balance's validity period has not been set, the PIN_FLD_VALID_FROM_DETAILS and PIN_FLD_VALID_TO_DETAILS fields are returned for that balance. These fields store three values in separate bits: the mode of the validity start and end times (such as first usage or relative), the relative offset unit, and the number of offset units in the relative period.

Note:

- A read/write transaction must be open before calling PCM_OP_BAL_GET_BALANCES. Otherwise, the object is locked, and the read operation returns an error.
- You can call PCM_OP_BAL_GET_BALANCES in a read-only operation, but the PIN_FLD_READ_BALGRP_MODE field in the input flist must be set to PIN_BAL_READ_BALGRP_CALC_ONLY.

Finding a Balance Group and Service for Bill Units

To retrieve a balance group and service for all the bill units in an account or for a single bill unit, BRM uses PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC.

To retrieve the balance group and service for a single bill unit, use PCM_OP_BAL_GET_BAL_GRP_AND_SVC.

These opcodes return the **/balance_group** and **/service** object POIDs based on the **/account** POID, **/billinfo** POID, and event end time (PIN_FLD_END_T) passed in the input flist. The balance groups for a specified bill unit do not have to be associated with a service yet to be returned.

If the CALC_ONLY flag is set, PCM_OP_BAL_GET_BAL_GRP_AND_SVC reads the balance groups and services from the cache instead of from the database.

The PIN_FLD_FLAGS field in the input flist controls the type of information returned in the output flist. The settings for this flag are:

- **PIN_BAL_GET_SERVICE_DEFAULT (0)**: do not return login name and balance group name (default).
- **PIN_BAL_GET_SERVICE_LOGIN (1)**: Return the login name and balance group name.
- **PIN_BAL_GET_SERVICE_ALIAS_LIST (2)**: Return the alias list for the service.
- **PIN_BAL_GET_DEFAULT_BAL_GRP (3)**: Return the default balance group of the bill unit.
- **PIN_BAL_GET_DEFAULT_GRP_AND_SVC (4)**: Return the default balance group of the bill unit and the default service of the default balance group.
- **PIN_ BAL_GET_ALL_BAL_GRP_AND_SVC (5)**: Return all balance groups, including balance groups not associated with a service, balance groups associated with a service, and the account default balance group.

> **Note:**
>
> You can also pass in flags to get the balance group name and service login aliases.

Balance groups not yet associated with a service are returned when the PIN_FLD_FLAGS value is not set or is set to **0**, **1**, or **2**. If the balance group returned is the account default balance group, the service object field in the output flist is NULL. If the balance group returned is *not* the account default balance group, there is no service object field in the output flist.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC over PCM_OP_BAL_GET_BAL_GRP_AND_SVC because it gives you the additional flexibility of retrieving balance groups and services for either multiple or single bill units.

Getting a List of Bills

PCM_OP_AR_GET_ACCT_BILLS retrieves the list of bills for all bill units in an account or for a single bill unit. The input flist determines the filter conditions to use for retrieving the data from the database.

You can restrict the search by various means, for example, date, status, and number of bills to be retrieved. You can also choose to find bills for the bill units, or for the bill units and their nonpaying child bill units.

PCM_OP_AR_GET_ACCT_BILLS returns the last bill. It uses the value in PIN_FLD_FLAGS to return the following:

- The latest bill (regular or corrective) when PIN_FLD_FLAGS is set to PIN_AR_LAST_BILL
- The previous (regular or corrective) bill only when PIN_FLD_FLAGS contains PIN_AR_ORIG_BILL
- All bills (the regular bill and all its corrective bills) when PIN_FLD_FLAGS contains PIN_AR_ALL_BILLS

The **pin_ar.h** file defines these constants as follows:

```
#define PIN_AR_LAST_BILL 0x01
#define PIN_AR_ORIG_BILL 0x02
#define PIN_AR_ALL_BILLS 0x04
```

PCM_OP_AR_GET_ACCT_BILL calls PCM_OP_AR_GET_BILLS to perform the search. For corrective bills, it provides the PIN_FLD_FLAGS as an input to PCM_OP_AR_GET_BILLS.

For corrective billing, PCM_OP_AR_GET_ACCT_BILLS optionally returns the following items in the PIN_FLD_RESULTS array:

- PIN_FLD_ORIG_NUM. The bill number from the previous bill.
- PIN_FLD_NAME. The name for the bill object. The **pin_bill.h** file contains the following definitions for corrective bills:

  ```
  #define PIN_OBJ_NAME_CORRECTIVE_BILL "PIN Corrective Bill"
  #define PIN_OBJ_NAME_CORRECTIVE_BILL_NOW "PIN Corrective Bill Now"
  #define PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND "PIN Corrective Bill On Demand"
  ```

- PIN_FLD_AMOUNT_ORIG. The PIN_FLD_DUE field for the previous bill.

- PIN_FLD_CREATED_T. The date that the bill was generated. It is taken from the corresponding field from the **/event/billing/corrective_bill** object.
- PIN_FLD_REASON_DOMAIN_ID. The reason domain code for the corrective bill taken from the corresponding **/event/billing/corrective_bill** object.
- PIN_FLD_REASON_ID. The reason ID for the corrective bill taken from the corresponding **/event/billing/corrective_bill** object.
- PIN_FLD_INV_TYPE. The invoice type for the bill.

Getting Bill Items

If general ledger (G/L) collection is enabled, PCM_OP_AR_GET_BILL_ITEMS retrieves the data from G/L **/journal** objects. Otherwise, this opcode retrieves the data from the events for each bill item. Using **/journal** objects improves performance.

The **pin_flds.h** file contains two decimal fields PIN_FLD_BILLED_AMOUNT and PIN_FLD_UNBILLED_AMOUNT used to store billed and unbilled amounts respectively.

For corrective bills, PCM_OP_AR_GET_BILL_ITEMS uses the input PIN_FLD_FLAGS value to select items that have unbilled and/or billed amounts allocated to them (where the allocated amount is greater than zero):

- When PIN_FLD_FLAGS is set to PIN_AR_BILLED_ITEM, PCM_OP_AR_GET_BILL_ITEMS retrieves the items with billed amounts that are allocated to the specified bill. The PIN_FLD_BILLED_AMOUNT elements in PIN_FLD_RESULTS output array returned by this code contain the billed amounts for the allocated items where the allocated amount is greater than zero. In addition, this opcode calculates the sum of the billed amounts that it placed in the PIN_FLD_RESULTS output array and returns that amount in a separate field called PIN_FLD_BILLED_AMOUNT.
- When PIN_FLD_FLAGS is set to PIN_AR_UNBILLED_ITEM, PCM_OP_AR_GET_BILL_ITEMS retrieves the items with unbilled amounts that are allocated to the specified bill. The PIN_FLD_UNBILLED_AMOUNT element in PIN_FLD_RESULTS output array returned by this code contain the unbilled amounts for the allocated items where the allocated amount is greater than zero. In addition, this opcode calculates the sum of the unbilled amounts that it placed in the PIN_FLD_RESULTS output array and returns that amount in a separate field called PIN_FLD_UNBILLED_AMOUNT.
- If PIN_FLD_FLAGS is not present or is present but does not have either value, PCM_OP_AR_GET_BILL_ITEMS retrieves all items allocated to the specified bill. The PIN_FLD_RESULTS output array returned by this code contains all the items allocated to the bill. However, this opcode does not return the summation of the billed or unbilled amounts.

Getting Bills

To get data for corrective bills, PCM_OP_AR_GET_BILLS uses the value in the PIN_FLD_FLAGS input field:

- It returns the latest bill (regular or corrective) when PIN_FLD_FLAGS contains PIN_AR_LAST_BILL. This is the default behavior.
- It returns the previous (regular or corrective) bill only when PIN_FLD_FLAGS contains PIN_AR_ ORIG_BILL.

- It returns all bills (the regular bill and all its corrective bills) when PIN_FLD_FLAGS contains PIN_AR_ALL_BILLS. The bills are returned in PIN_FLD_RESULTS sorted by the time when they were created.
- PCM_OP_AR_GET_BILLS optionally returns the following items in the PIN_FLD_RESULTS array:
  - PIN_FLD_ORIG_NUM. The bill number from the previous bill.
  - PIN_FLD_NAME. The name for the bill object. The **pin_bill.h** file contains the following definitions for corrective bills:

    ```
    #define PIN_OBJ_NAME_CORRECTIVE_BILL "PIN Corrective Bill"
    #define PIN_OBJ_NAME_CORRECTIVE_BILL_NOW "PIN Corrective Bill Now"
    #define PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND "PIN Corrective Bill On Demand"
    ```

  - PIN_FLD_AMOUNT_ORIG. The PIN_FLD_DUE field for the previous bill.
  - PIN_FLD_CREATED_T. The date that the bill was generated. It is taken from the corresponding field from the **/event/billing/corrective_bill** object.
  - PIN_FLD_REASON_DOMAIN_ID. The reason domain code for the corrective bill taken from the corresponding **/event/billing/corrective_bill** object.
  - PIN_FLD_REASON_ID. The reason id for the corrective bill taken from the corresponding **/event/billing/corrective_bill** object.
  - PIN_FLD_INV_TYPE. The invoice type for the bill.

Finding Items

To find items, BRM uses PCM_OP_PYMT_ITEM_SEARCH. This opcode searches for **/item** objects with a variable number of input parameters. It is called by other opcodes for handling adjustments and disputes.

- PCM_OP_PYMT_ITEM_SEARCH takes a variable number of arguments to restrict the scope of the search.
- The PIN_FLD_POID field is mandatory and is a dummy POID used for getting the database ID for the search.
- The bill unit field is optional. This field specifies the bill unit that owns the items. You can use either the **/billinfo** POID or the **/billinfo** ID. If a bill unit is not specified, the bill unit associated with the account's default balance group is selected.
- You can specify either PIN_FLD_ACCOUNT_OBJ or PIN_FLD_ACCOUNT_NO to narrow the search to a particular account.
- For collective bills, if they are not located in the bill table, PCM_OP_PYMT_ITEM_SEARCH checks the **RejectPaymentForPreviousBill** business parameter.
  - If **RejectPaymentForPreviousBill** is enabled to prevent BRM from accepting the payment, the opcode sends the payment to the suspense payment account.
  - If the **RejectPaymentForPreviousBill** is disabled to prevent BRM from accepting the payments, the opcode will search for the bill in the **history_bills** objects. If the bill is located in the **history_bills** objects, the payment is accepted for the bill with the same POID as the bill being processed. If such a bill is not located in **history_bills** objects, the payment is sent to a suspense payment account.
- If PIN_FLD_STATUS, PIN_FLD_BILL_OBJ, or both are specified, the search matches the items equal to these values.

- PIN_FLD_START_T and PIN_FLD_END_T specify the time range for the search. These arguments narrow the search to items whose PIN_FLD_CREATED_T value falls between the start and end times.
- The PIN_FLD_POID field in the PIN_FLD_ARGS array narrows the search to specific ***/item*** object types. This is a TYPE_ONLY POID. Different items have different POID object types.

  For example, to return only the cycle forward and usage items, create the POID for the PIN_FLD_POID field as follows:

```
PIN_POID_CREATE(database,
      " '/item/cycle_forward', '/item/usage' " -1, ebufp);
```

Finding Discounts in Bill Items

BRM uses PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT to retrieve the discount for events of a given bill item.

For each event it retrieves, this opcode calculates the total amount of each balance and the total discount amount of each balance.

If a bill has been corrected and the database contains a corrective bill for the bill unit, BRM retrieves the events associated with the corrective bill only. It does not retrieve the events for the prior bill.

Retrieving a Balance Summary

To retrieve the balance summary from an account, BRM uses PCM_OP_AR_GET_ACCT_BAL_SUMMARY. This opcode retrieves the unapplied, open bill due, pending bill due, and total dispute balances for all the bill units in a specified account or for a particular bill unit.

You can use PCM_OP_AR_GET_BAL_SUMMARY to get this information for a single bill unit.

For both opcodes, you can specify whether BRM returns the balance for:

- The specified bill units (for PCM_OP_AR_GET_BAL_SUMMARY, this is a single bill unit).
- The specified bill units and their nonpaying child bill units (for PCM_OP_AR_GET_BAL_SUMMARY, this is a single bill unit).

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_BAL_SUMMARY over PCM_OP_AR_GET_BAL_SUMMARY because it gives you the additional flexibility of retrieving balance summaries for either multiple or single bill units.

PCM_OP_AR_GET_ACCT_BAL_SUMMARY retrieves the following data for the specified bill unit:

> **Note:**
>
> The data includes the amounts of the bill unit's nonpaying child bill units if you set the appropriate flag.

- Unapplied amount.

- Amount due for the open bill.
- Amount due for the pending bill.
- Total disputed amount.
- For PCM_OP_AR_GET_BAL_SUMMARY, the Bill in Progress information when there are pending items and the balance is greater than or equal to 0 because of open disputes against the items.

You can use the following PCM_OP_AR_GET_BAL_SUMMARY input criteria for retrieving the balance:

- The POID of the A/R bill unit object or *paying* bill unit object.
- The POID of the bill unit for which to retrieve the balance summary.
- A flag indicating whether to retrieve the balances for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return bills for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN flag to **0**.
  - To return bills for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN flag to **1**.
- For PCM_OP_AR_GET_BAL_SUMMARY, whether to retrieve Bill in Progress information if there are any pending items but the balance is zero.

  To do so, PCM_OP_AR_GET_BAL_SUMMARY uses the PIN_FLD_ITEM_PENDING_FLAGS flag. If this flag is set to **1** in the output flist, the opcode provides access to Bill in Progress information on the balance summary. If is flag is set to **0**, the opcode does not provide access to Bill in Progress information.

Retrieving a List of Bills for a Bill Unit

BRM uses PCM_OP_AR_GET_ACCT_BILLS to get a list of bills for all the bill units in a specified account or for a particular bill unit. The input flist determines the filter conditions to use for retrieving the data from the database.

You can restrict the search by various means, for example, date, status, and number of bills to be retrieved. You can also choose to find bills for the bill units, or for the bill units and their nonpaying child bill units.

You can use PCM_OP_AR_GET_BILLS to get this information for a single bill unit. You can restrict the search by various means, for example, date, status, and number of bills to be retrieved. You can also choose to find bills for the specific bill unit, or for it and its nonpaying child bill units.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_BILLS over PCM_OP_AR_GET_BILLS because it gives you the additional flexibility of retrieving the bill list for either multiple or single bill units.

About the Bill Data Retrieved

PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS retrieve a list of bills for the account's bill units or it retrieves the bill units and their nonpaying child bill units. PCM_OP_AR_GET_BILLS retrieves only the bill unit you specify in the input flist. For each bill, the output includes the following data:

- Current and previous total

- Total for nonpaying child bill units, provided you set the appropriate flag
- Amounts for adjustments, disputes, transfers, write-offs, and so on
- The bill's state

Specifying the Search Criteria for Retrieving Bills

When you use PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS, you can use the following input criteria for retrieving the list of bills:

- The POID of the account, bill unit, or bill object.

  These opcodes list a specific bill if the bill object POID is passed in the input flist. If you specify an account POID for PCM_OP_AR_GET_ACCT_BILLS, the opcode retrieves the bills for all the bill units in the account.

- The POID of the A/R bill unit.

  > **Note:**
  >
  > – For PCM_OP_AR_GET_ACCT_BILLS, this field is mandatory if the input flist specifies the bill unit or bill POID instead of the account POID.
  >
  > – For PCM_OP_AR_GET_BILLS, this field is mandatory unless you include the POID of the A/R account.

- The POID of the A/R account. This returns the default bill unit for the A/R account.

  > **Note:**
  >
  > This field is not available for PCM_OP_AR_GET_ACCT_BILLS.

- A flag indicating whether to retrieve the balances for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return balances for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return balances for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- The status of the bills to retrieve. If the PIN_FLD_STATUS flag is set to:
  - **2**: The opcode retrieves open bills.
  - **4**: The opcode retrieves closed bills.
  - **6**: The opcode retrieves all bills.
- The number of bills to retrieve. If you use PCM_OP_AR_GET_ACCT_BILLS to retrieve bills from all the bill units in an account, this is the number of bills the opcode retrieves from each bill unit. It is *not* the total number of bills to retrieve from the account unless the account has only one bill unit.
- The start and end times for the bills to retrieve.

- The sort order, ascending or descending. If you use PCM_OP_AR_GET_ACCT_BILLS to retrieve bills from all the bill units in an account, this is the sort order the opcode applies to the bills in each bill unit, *not* across all the bill units.

Retrieving A/R Items That Apply to a Bill Unit

BRM uses PCM_OP_AR_GET_ACCT_ACTION_ITEMS to get a list of A/R items for all the bill units in a specified account or for a particular bill unit.

You can restrict the search by various means, such as date, status, and bill unit POID.

You can find items for the specific bill unit or for it and its nonpaying child bill units.

The A/R item types returned can be:

- **/item/adjustment**
- **/item/dispute**
- **/item/payment**
- **/item/refund**
- **/item/writeoff**

Items associated with the A/R items, such as, **/item/payment/reversal**, **/item/adjustment_reversal**, and **/item/settlement**, are returned in the PIN_FLD_RELATED_ACTION_ITEM_OBJ field.

You can use PCM_OP_AR_GET_ACTION_ITEMS to get this information for a single bill unit.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_ACTION_ITEMS over PCM_OP_AR_GET_ACTION_ITEMS because it gives you the additional flexibility of retrieving the A/R items for either multiple or single bill units.

About the Item Data Retrieved

PCM_OP_AR_GET_ACCT_ACTION_ITEMS and PCM_OP_AR_GET_ACTION_ITEMS return the following data:

> **Note:**
>
> PCM_OP_AR_GET_ACCT_ACTION_ITEMS returns an array. If the PIN_FLD_POID field is the account POID in the input flist, each element in the array is one of the listed fields for a bill unit in the account. Otherwise, the returned fields are for the requested single bill unit.

- The **/billinfo** and A/R bill unit object (PIN_FLD_BILLINFO_OBJ and PIN_FLD_AR_BILLINFO_OBJ) that the item belongs to.
- The account's primary contact information (PIN_FLD_NAMEINFO array), such as company, first name, last name, middle name, and salutation.
- The number of A/R items that were found (PIN_FLD_THRESHOLD) *only* if this count exceeds the threshold specified in the input. If the specified threshold is **not** exceeded, all the A/R items found are returned.

- Either the A/R items for *only* the given bill unit or the A/R items for the given bill unit *and* its nonpaying child bill units, depending on the flag setting in the input flist.
- For payments and reversals (that apply):
  - The POID of the reversal item (PIN_FLD_RELATED_ACTION_ITEM_OBJ).
  - The revision time when the payment was reversed; that is, when the reversal was posted (PIN_FLD_POSTED_T).
- For disputes and settlements (that apply):
  - The POID of the settlement item (PIN_FLD_RELATED_ACTION_ITEM_OBJ).
  - The revision time when the dispute was settled; that is, when the settlement was posted (PIN_FLD_POSTED_T).
  - The bill item to which the dispute or settlement is made (PIN_FLD_RELATED_BILL_ITEM_OBJ).
- For PCM_OP_AR_GET_ACTION_ITEMS only:
  - For event adjustments, disputes, and settlements, PCM_OP_AR_GET_ACTION_ITEMS returns a PIN_FLD_AGGREGATE_AMOUNTS array for each balance affected by the adjustment, dispute, or settlement item. This array contains the aggregated amount of adjustments, disputes, and settlements for each balance.
  - For adjustments, disputes, and settlements, PCM_OP_AR_GET_ACTION_ITEMS indicates whether the adjustment, dispute, or settlement was created for an event or for an item. PCM_OP_AR_GET_ACTION_ITEMS also indicates whether the adjustment, dispute, or settlement affects single or multiple balances.
  - For adjustments, PCM_OP_AR_GET_ACTION_ITEMS reads both **/item/adjustment** and **/item/adjustment_reversal** for a bill unit. It filters out **/item/adjustment_reversal** from the output and adds the POID of **/item/adjustment_reversal** as PIN_FLD_RELATED_ACTION_ITEM_OBJ and then adds PIN_FLD_REVERSED in the output.

For possible return values of each A/R action retrieved, refer to the fields contained in the PIN_FLDS_RESULTS array in the output flist specification.

Specifying Search Criteria for Retrieving Items

PCM_OP_AR_GET_ACCT_ACTION_ITEMS and PCM_OP_AR_GET_ACTION_ITEMS take as input:

- The POID of the A/R bill unit (PIN_FLD_AR_BILLINFO_OBJ).
- The POID of the bill unit related to the A/R items.

  When retrieving A/R items for nonpaying child bill units, the bill unit object is that of the paying bill unit. Otherwise, it is the bill unit object of the specified bill.
- The POID of the bill unit for which to retrieve A/R items (PIN_FLD_AR_BILLINFO_OBJ). This field is required only when retrieving A/R items for a specific, nonpaying child bill unit.
- A flag indicating whether to retrieve the A/R items for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return A/R items for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return A/R items for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.

- The maximum number, or threshold, of A/R items to retrieve.

  > **Note:**
  >
  > – The threshold field is valid only for an A/R bill unit (that is, the paying bill unit of a parent account).
  >
  > – If you use PCM_OP_AR_GET_ACCT_ACTION_ITEMS to retrieve A/R items from all the bill units in an account, this is the maximum number of A/R items the opcode retrieves from each bill unit. It is not the total number of A/R items retrieved from the account unless the account has only one bill unit.

- The action-item type to retrieve (for example, **/item/adjustment**) specified as a comma-delimited string.

  > **Note:**
  >
  > When retrieving only payments, both payments and reversals are specified. When retrieving only disputes, both disputes and settlements are specified.

- The time range for which to retrieve A/R items. The start date is inclusive and the end date is exclusive.
- The amount range. The from and to amounts are inclusive.
- The amount indicator for whether to retrieve credits or debits for the given range.
- The item status. You can query based on the status of open, closed, or reversed (valid for payments only).

  > **Note:**
  >
  > There is no *pending* status for A/R items.

Retrieving a List of Bill Items for a Bill Unit

To retrieve a list of bill items for a bill unit, BRM uses PCM_OP_AR_GET_BILL_ITEMS.

You can restrict the search by various means (for example, date, status, and bill unit POID). You can also choose to find items for the specific bill unit or for it and its nonpaying child bill units.

Depending on the value of the following CM **pin.conf** entry, the charge and discount for the items are populated in the PCM_OP_AR_GET_BILL_ITEMS output flist. The value for this entry can be either **0** or **1**. The default is **0**.

```
- fm_ar skip_displaying_charge_and_discount_for_item 1
```

If the value specified is **1**, the charge and discount for the items is not populated in the PCM_OP_AR_GET_BILL_ITEMS output flist.

> **Note:**
>
> PCM_OP_AR_GET_BILL_ITEMS is used by Billing Care and Customer Center to display the item details. Setting this flag to **1** skips the query to get the charge and discount from the events belonging to the item, thereby improving performance.

About the Bill Item Data Retrieved

PCM_OP_AR_GET_BILL_ITEMS returns the following:

- The bill items for only the specified bill unit if PIN_FLD_INCLUDE_CHILDREN is set to **0**.
- The bill items for the specified bill unit and its nonpaying child bill units if PIN_FLD_INCLUDE_CHILDREN is set to **1**.
- The bill items for the nonpaying child bill units if PIN_FLD_INCLUDE_CHILDREN is set to **2**.
- The number of bill items that were found (PIN_FLD_THRESHOLD) *only* if this count exceeds the threshold specified in the input. If the specified threshold is *not* exceeded, all the bill items found are returned.

For each bill item retrieved, the fields in the PIN_FLDS_RESULTS array are returned.

If general ledger (G/L) collection is enabled, PCM_OP_AR_GET_BILL_ITEMS retrieves the data from G/L **/journal** objects. Otherwise, the opcode retrieves the data from the events for each bill item. Using **/journal** objects improves performance.

Customizing the Number of Items to Retrieve in a Search

PCM_OP_AR_GET_BILL_ITEMS uses step-search. The step size (the number of items it tries to retrieve simultaneously) is determined by the following CM **pin.conf** entry:

```
- fm_bill item_search_batch 10000
```

Specifying Search Criteria for Bill Items

PCM_OP_AR_GET_BILL_ITEMS takes as input:

- The POID of the A/R bill unit or *paying* bill unit object (PIN_FLD_AR_BILLINFO_OBJ).
- The POID of the bill object related to the bill items.

  When PCM_OP_AR_GET_BILL_ITEMS retrieves bill items for nonpaying child bill units, the bill object is that of the paying bill unit object. Otherwise, it is the bill object of the specified bill unit.
- The POID of the bill unit object (PIN_FLD_BILLINFO_OBJ) for which to retrieve bill items. This field is required only when retrieving bill items for a specific, nonpaying child bill unit.
- A flag indicating whether to retrieve the bill items for only the specified bill unit or for the specified bill unit and its nonpaying child bill units:
  - To return bill items for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return bill items for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.

  - To return bill items for the nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **2**.
- The service device ID or phone number or login name to retrieve the SERVICE_OBJ for the services with matching ALIAS or LOGIN_NAME. SERVICE_OBJ is used to retrieve the associated bill items.
- The maximum number, or threshold, of bill items to retrieve.
- The time range in which to retrieve bill items. The start date is inclusive and the end date is exclusive.
- The amount range. The from and to amounts are inclusive.
- The amount indicator for whether to retrieve credits or debits for the given range.
- The item status. You can query based on the status of open, closed, or pending.

Retrieving Details about a Specific A/R Item or Bill Item

You can retrieve details for an A/R item or bill item in a bill unit using either of two opcodes:

- PCM_OP_AR_GET_ITEMS: Use this opcode to retrieve the details of an A/R item or bill item for a bill unit. For adjustments, disputes, and settlements, this opcode retrieves the aggregated impact amount for each balance or the events associated with the item (**/item/dispute**, **/item/adjustment**, **/item/adjustment_reversal**, and so on). PCM_OP_AR_GET_ITEMS provides details for both currency and noncurrency balances.
- PCM_OP_AR_GET_ITEM_DETAIL: Use this opcode to retrieve the details of an A/R item or bill item for a bill unit. This opcode retrieves detailed information about a specific write-off or usage item. For reversed adjustments, it sends the POID of **/item/adjustment_reversal** as PIN_FLD_RELATED_ACTION_ITEM_OBJ and returns the reversed amount in the PIN_FLD_REVERSED field. It does *not* return the aggregated dispute amount for each balance for disputed events. More importantly, it does *not* return details about noncurrency balances for adjustments, disputes, or settlements unless the item has both a currency *and* noncurrency component.

You should base your choice of opcode on how much information you want to retrieve for the A/R items and bill items.

About the Item Data Retrieved

PCM_OP_AR_GET_ITEMS calls PCM_OP_AR_GET_ITEM_DETAIL to get the details of the A/R item or bill item. When returning details for adjustments, disputes, or settlements, it collects details for:

- Item adjustments, disputes, and settlements (**/event/billing/dispute/item**, **/event/billing/adjustment/item**, and so on).
- Event adjustments, disputes, and settlements (**/event/billing/dispute/event**, **/event/billing/adjustment/event**, and so on).

Depending on the contents of the input flist, PCM_OP_AR_GET_ITEMS takes one of the following actions:

- If the input flist contains the POID of a transfer event, PCM_OP_AR_GET_ITEMS calls PCM_OP_AR_GET_ITEM_DETAIL to get the details of the transfer and performs no further action. For information on what this opcode returns for transfer events, see "About the Item Detail Data Retrieved".
- If the input flist contains the POID of a bill item or an A/R action such as an adjustment, dispute, or settlement, PCM_OP_AR_GET_ITEMS calls

PCM_OP_AR_GET_ITEM_DETAIL to get the details of the A/R action or bill item. It also calls PCM_OP_AR_RESOURCE_AGGREGATION to calculate and return the aggregated amount of any event adjustments, disputes, and settlements for each balance.

For A/R actions, it determines whether the action was initiated at the item level or event level. If the action was initiated at the event level, it retrieves both currency and noncurrency balances for the event.

For information on PCM_OP_AR_RESOURCE_AGGREGATION, see "Retrieving Details on Available Balances".

PCM_OP_AR_GET_ITEMS returns an array of details for the A/R action or bill item. The **transfers_into** array provides the details of all items that have a currency or noncurrency impact for the A/R item or bill item specified in the input flist. These details include the aggregated balance impacts for each event. The **AGGREGATE_AMOUNTS** array returns the PIN_FLD_REVERSED field if the corresponding adjustment is reversed.

For adjustments, disputes, and settlements, the details for each A/R action include a flag in the transfer array that indicates whether the action was initiated at the event level or item level:

- PIN_FLD_ADJUSTMENT_TYPE: Set to **0** for item adjustments or **1** for event adjustments.
- PIN_FLD_DISPUTE_TYPE: Set to **0** for item disputes or **1** for event disputes.
- PIN_FLD_SETTLEMENT_TYPE: Set to **0** for item settlements or **1** for event settlements.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for any adjustments, disputes, or settlements. If this flag is set to **0**, the action affects one balance only. If the flag is set to **1**, the action affects several different balances (for example, multiple currencies or both a currency and a noncurrency balance, such as minutes).

About the Item Detail Data Retrieved

PCM_OP_AR_GET_ITEM_DETAIL retrieves details of activities that had a currency impact for the specified A/R item or bill item. If the input flist contains the POID for the transfer event, that POID is used to retrieve the item details from the transfer event binary large object (BLOB) file. PCM_OP_AR_GET_ITEM_DETAIL returns an array of the bill items the transfer event affected and the input POID.

- The **transfers_into** array returns the details of all the items that received some currency amount from the item specified in the input flist. For example, when a customer service representative (CSR) opens a dispute for a subscription fee, money is transferred into the dispute item from the subscription bill item (the cycle forward item). When the cycle forward item is passed in the input flist, the details of all the disputes that received money from this bill item are returned in the **transfers_into** array.
- The **transfers_out** array returns the details of all those items that transferred some currency amount to the item specified in the input flist. For example, when a customer pays for service usage, money is transferred from the payment item to the bill item (usage item). When the usage item is passed in the input flist, the details of all the payments that transferred money to this bill item are returned in the **transfers_out** array. PCM_OP_GET_ITEM_DETAILS opcode will be modified to show settled tax, settled tax_amount amount information based on taxcode and jurisdiction level for each item in the transfers_out array of the output flist. settled_non_taxed_amount will be consolidated amount shown for each item. If **tax_return_juris** is not set as itemize, then settled tax information for each item will be displayed based on tax code only.

For adjustments, disputes, and settlements, the details for the A/R action include a flag in the transfer array that indicates whether the action was initiated at the event level or item level:

- PIN_FLD_ADJUSTMENT_TYPE: Set to **0** for item adjustments or **1** for event adjustments.

- PIN_FLD_DISPUTE_TYPE: Set to **0** for item disputes or **1** for event disputes.
- PIN_FLD_SETTLEMENT_TYPE: Set to **0** for item settlements or **1** for event settlements.

PCM_OP_AR_GET_ITEM_DETAIL determines how to set this flag by checking to see whether the event is stored in an **/event/billing/***AR_action_type***/item** object or an **/event/billing/***action_type***/event** object. The opcode returns the following information:

- **Item adjustments, disputes and settlements**: Returns a transfer array that includes details of the adjustment, dispute, or settlement item in the PIN_FLD_EVENT_OBJ field. In this case, the only event PCM_OP_AR_GET_ITEM_DETAIL looks at is the single event associated with the **/item** object. For example, it looks at the single **/event/billing/dispute/item** event associated with the **/item/dispute** object.
- **Event adjustments, disputes and settlements**: Returns a transfer array that includes a PIN_FLD_EVENTS array with the details of all the events that make up the adjustment, dispute, or settlement. In this case, PCM_OP_AR_GET_ITEM_DETAIL looks at all the events identified in the **/item** object.
- PIN_FLD_SETTLED_TAX, PIN_FLD_SETTLED_TAXED_AMOUNT, PIN_FLD_SETTLED_NON_TAXED_AMOUNT: Returns information about the settled tax amount, the settled taxed amount, and the settled non-taxed amount at the item, bill, and event level, for disputes and settlements.

The details for each A/R action also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for any adjustments, disputes, or settlements. If this flag is set to **0**, the action affected one balance only. If the flag is set to **1**, the action affected several different balances (for example, multiple currencies or both a currency and a noncurrency balance, such as minutes).

In addition to retrieving the other details of the adjustment, dispute, or settlement, PCM_OP_AR_GET_ITEM_DETAIL determines whether there is a tax reversal already calculated for the event. If so, it adds this information to the PIN_FLD_TAX field in the output flist.

> **Note:**
>
> Tax reversals are calculated at adjustment, dispute, or settlement event creation time only if you have set up tax now as the tax method for adjustments, disputes, and settlements in the CM **pin.conf** file.

The details for each A/R action also include a PIN_FLD_RESOURCE_IMPACTED flag to indicate whether the action affects a single balance (**0**) or multiple balances (**1**).

Specifying Search Criteria for Retrieving Items

PCM_OP_AR_GET_ITEMS and PCM_OP_AR_GET_ITEM_DETAIL take as input:

- The POID of the item object or the POID of the transfer event. If the POID is for a transfer event, all other values are ignored.
- The POID of the disputed item (PIN_FLD_RELATED_BILL_ITEM_OBJ) if the bill item is disputed. This field is mandatory if the item object is a dispute, but you can also include this field for other types of item objects.
- The POID of the settlement item or the reversal item (PIN_FLD_RELATED_ACTION_ITEM_OBJ) if the item is for a dispute or payment *and* a dispute settlement or payment reversal exists.

About the Item Tax Detail Retrieved

PCM_OP_AR_GET_ITEM_DETAIL returns details about the taxes.

- **For item-level disputes and settlements**:
  - If the PIN_AR_WITH_TAX flag is passed in, the opcode calculates the tax on the dispute amount.
  - If the **ItemizedTaxCalculation** business parameter is enabled, the opcode calculates the proportional tax for each item. The disputed tax amount is updated in the output flist's PIN_FLD_TAX.PIN_FLD_AMOUNT_TAX_ADJUSTED field. For information about setting the business parameter, see "Configuring the Default Tax Method for Account Adjustments" in *BRM Calculating Taxes*.
  - If **TaxReturnJuris** is enabled (set to **itemize**), you can see information based on taxcode and jurisdiction level for each item in the transfers_out array of the output flist. The settled_tax_amount, settled_taxed_amount, and settled_non_taxed_amount will be shown as a consolidated amount for each item if it is set to **summarize**(default).

> **Note:**
>
> For settled disputes, no tax information will be shown. The PIN_FLD_RELATED_ACTION_ITEM_OBJ field will be updated with its corresponding settlement.

- **For account-level adjustments**: If the **tax_now** CM **pin.conf** entry is set to **3**, the opcode calculates the tax proportionally. The opcode creates a new **/event/billing/adjustment/account** to record the tax impact. For information about setting the CM **pin.conf** entry, see "Configuring the Default Tax Method for Account Adjustments" in *BRM Calculating Taxes*.
- **For bill and item-level adjustments**: If the **ItemizedTaxCalculation** business parameter is enabled, the adjusted tax amount is updated in the output flist's PIN_FLD_TAXES.PIN_FLD_AMOUNT_TAX_ADJUSTED field. For information about setting the business parameter, see "Configuring Itemized Taxation for Adjustments and Payments" in *BRM Calculating Taxes*.

Retrieving Dispute Details for a Bill Unit

You can retrieve dispute details for a bill unit by using either of the following opcodes:

- To return a complete set of dispute details, use PCM_OP_AR_GET_DISPUTE_DETAILS. This opcode retrieves the details of all event and item disputes for a bill unit and the aggregated disputed amount for the events associated with a dispute item (**/item/dispute** object).

  See "Retrieving a Full Set of Dispute Data".

- To return a limited set of dispute details, use PCM_OP_AR_GET_DISPUTES. This opcode retrieves details of event and item disputes for a bill unit. This opcode does not return as much information as PCM_OP_AR_GET_DISPUTE_DETAILS. For example, it does *not* return the aggregated disputed amount for each balance.

  See "Returning a Limited Set of Dispute Data".

You should base your choice of opcode on how much information you want to retrieve for the disputes.

Retrieving a Full Set of Dispute Data

PCM_OP_AR_GET_DISPUTE_DETAILS calls PCM_OP_AR_GET_DISPUTES to get the details of each dispute. It collects details for both item disputes (**/event/billing/dispute/item**) and event disputes (**/event/billing/dispute/event**).

When it has the details for each dispute, PCM_OP_AR_GET_DISPUTE_DETAILS creates an input flist that contains the A/R event POID of each dispute and passes it to PCM_OP_AR_RESOURCE_AGGREGATION. PCM_OP_AR_RESOURCE_AGGREGATION calculates and returns the aggregated disputed amount for each balance. Balances can be currency or noncurrency. For information on PCM_OP_AR_RESOURCE_AGGREGATION, see "Retrieving Details on Available Balances".

Data Retrieved by PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES returns an array of disputes. This array includes transfer arrays identifying the **/item/dispute** object for each event and item dispute. In addition, the arrays for the event disputes include an aggregated balance subarray.

Each dispute includes a PIN_FLD_DISPUTE_TYPE flag in the transfer array to indicate whether it was filed as an event dispute or an item dispute:

- If this flag is set to **0**, the dispute is an item dispute.
- If this flag is set to **1**, the dispute is an event dispute.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for the disputes. If this flag is set to **0**, the action affected one balance only. If the flag is set to **1**, the action affected several different balances (for example, multiple currencies or both a currency and a noncurrency balance, such as minutes).

Specifying Search Criteria for PCM_OP_AR_GET_DISPUTE_DETAILS

PCM_OP_AR_GET_DISPUTES takes as input:

- The POID of the bill unit that has a dispute. If the account object is specified, the default account bill unit is selected.
- The POID of the A/R bill unit or *paying* bill unit object (PIN_FLD_AR_BILLINFO_OBJ).
- A flag indicating whether to retrieve the dispute details for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return dispute details for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return dispute details for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- A status flag indicating whether to retrieve open disputes, closed (settled) disputes, or both. If the PIN_FLD_STATUS flag is set to:
  - **2**: The opcode retrieves open disputes.
  - **4**: The opcode retrieves settled disputes.
  - **6**: The opcode retrieves all disputes, open or settled.

Returning a Limited Set of Dispute Data

PCM_OP_AR_GET_DISPUTES calls PCM_OP_AR_GET_ACTION_ITEMS to get a list of disputes for the bill unit and passes each disputed item to PCM_OP_AR_GET_ITEM_DETAIL. PCM_OP_AR_GET_ITEM_DETAIL gathers the details on the disputed items. PCM_OP_AR_GET_DISPUTES then creates an output flist that provides the information transferred by PCM_OP_AR_GET_ITEM_DETAIL, including flags to indicate whether the dispute is an event dispute or an item dispute.

You can restrict the search by various means, for example, date, status, and bill unit POID. You can choose to find items for the specific bill unit, or for it and its nonpaying child bill units.

For open disputes, PCM_OP_AR_GET_DISPUTES retrieves unsettled disputes; for resolved disputes, it retrieves settled disputes.

Data Retrieved by PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES returns an array of disputes. Each dispute includes a PIN_FLD_DISPUTE_TYPE flag in the transfer array to indicate whether it was filed as an event dispute or an item dispute:

- If this flag is set to **0**, the dispute is an item dispute.
- If this flag is set to **1**, the dispute is an event dispute.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for the disputes. If this flag is set to **0**, the action affected one balance only. If the flag is set to **1**, the action affected several different balances (for example, multiple currencies or both a currency and a noncurrency balance, such as minutes).

Specifying Search Criteria for PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES takes as input:

- The POID of the bill unit that has a dispute. If the account object is specified, the default account bill unit is selected.
- The POID of the A/R bill unit or *paying* bill unit object (PIN_FLD_AR_BILLINFO_OBJ).
- A flag indicating whether to retrieve the dispute details for only the given bill unit or for the given bill unit and its nonpaying child bill units.
  - To return dispute details for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return dispute details for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- A status flag indicating whether to retrieve open disputes, closed (settled) disputes, or both. If the PIN_FLD_STATUS flag is set to:
  - **2**: The opcode retrieves open disputes.
  - **4**: The opcode retrieves settled disputes.
  - **6**: The opcode retrieves all disputes, open or settled.

Retrieving Details on Available Balances

BRM uses PCM_OP_AR_RESOURCE_AGGREGATION to retrieve the available balances for the events associated with a bill item. This opcode aggregates (combines) the balance impacts of all the events by balance, such as US dollars or included minutes.

If an adjustment, dispute, or settlement is associated with an event, this opcode also includes the dispute, adjustment, reversed adjustment, or settlement amount in the aggregated impacts for each balance. The balance types can include currency, noncurrency, or both. The output flist PIN_FLD_RESULTS array contains the remaining available amount for opening new adjustments, disputes, or settlements for each balance in the PIN_FLD_ALLOCATED field.

You can write a custom application to use the aggregated amounts to find the balance impact of an event and help determine how much of each balance for an event is available for A/R activities like adjustments. This opcode is also called by PCM_OP_AR_GET_DISPUTE_DETAILS and PCM_OP_AR_GET_ITEMS.

PCM_OP_AR_RESOURCE_AGGREGATION returns an array that lists the following information for each balance impacted by the events specified in the input flist:

> **Note:**
>
> If PCM_OP_AR_RESOURCE_AGGREGATION is returning adjustment, dispute, or settlement events, it includes a balance array for all the balances impacted by the event, whether or not a particular balance is impacted by the adjustment, dispute, or settlement. For example, given a currency adjustment of an event that impacts US dollars and minute balances, the opcode returns arrays for both balances, even though the adjustment did not affect minutes.

- Balance element ID.
- Aggregated event charges.
- If the balance was discounted, the discount amount.
- If the balance was adjusted, the adjustment amount.
- If an adjustment was reversed, the reversed amount and the amount of the original adjustment.
- If the balance was disputed, the dispute amount and the amount of the original dispute.
- If a settlement occurred, the settlement amount.
- The total balance currently available (PIN_FLD_ALLOCATED field). This is the amount in the PIN_FLD_AMOUNT field minus any discounts, adjustments, disputes (the dispute amount in PIN_FLD_DISPUTED only, *not* the original dispute amount), and settlements.

Finding Events Associated with an Account

To find all events associated with an account, BRM uses PCM_OP_BILL_POL_EVENT_SEARCH. This opcode is not called by any opcode.

For filtered searches, PCM_OP_BILL_POL_EVENT_SEARCH supports searching on the following criteria:

- POID of the bill unit

- POID of the bill
- POID of the item
- Date ranges
- Amount ranges
- Event type
- Service type
- Balance element ID

To specify the number of events to return, use a value for the threshold. If the threshold value is greater than or equal to the number of events, all events are returned. If the threshold value is less than the number of events, the number of events is returned.

By default, PCM_OP_BILL_POL_EVENT_SEARCH returns all the events for the account but discards dispute, adjustment, and settlement events. This opcode can be customized to retrieve all the events for the account and keep the dispute, adjustment, and settlement events.

Finding Events Associated with Bill Items

To find events associated with a bill item, BRM uses PCM_OP_BILL_ITEM_EVENT_SEARCH.

This opcode searches the **/event** object for details related to a specific item. This opcode retrieves a list of events for a given item POID and flag.

The PIN_FLD_SEARCH_TYPE field in the PCM_OP_BILL_ITEM_EVENT_SEARCH input flist can be set to one or all of the following fields:

- PIN_ITEM_EVENT_OWNED: Events owned by an item. If PIN_ITEM_EVENT_OWNED is selected, PCM_OP_BILL_ITEM_EVENT_SEARCH returns a list of events whose item POID matches the item POID passed in the input flist.
- PIN_ITEM_EVENT_SPONSORED: Events sponsored by the item. If the PIN_ITEM_EVENT_SPONSORED flag is selected, PCM_OP_BILL_ITEM_EVENT_SEARCH returns a list of events whose item POID in the event subtotal array matches the item POID passed in the input flist.
- PIN_ITEM_EVENT_TRANSFERRED: A/R events that affect the item's total. If the PIN_ITEM_EVENT_TRANSFERRED flag is selected, PCM_OP_BILL_ITEM_EVENT_SEARCH returns a list of events whose item POID matches POIDs stored in PIN_OBJ_TYPE_EVENT_ITEM_TRANSFER. Lists are returned in separate arrays so that a list can be returned for each selected flag.

In addition to the above criteria, the event search can be narrowed to reflect the type of event and when the event was created.

Performing Rollover Transfers

BRM performs rollover transfers by using PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER for each account or service that has a rollover-transfer profile associated with it.

PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER checks the rollover-transfer profile object, **/profile/rollover_transfer**, to make sure it is configured and valid for the balance and receiver and then transfers the entire rollover amount to the receiver.

BRM performs a rollover transfer as follows:

1. The BRM event notification system listens for the following events to occur:
   - **/event/billing/cycle/rollover/monthly**
   - **/event/billing/cycle/rollover_correction**
2. When one of the events occurs, the event notification system calls PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER.
3. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER checks the rollover-transfer profile object (**/profile/rollover_transfer**) to make sure it is configured and valid for the balance and the receiver.
4. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER transfers the rollover amount to the receivers' accounts.
5. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER removes the rolled-over amount from the sender's sub-balance and creates new sub-balances for the receivers with the transferred amount.

   The validity of the new sub-balance is based on the accounting cycle of the receiver and is valid for the remainder of the current accounting cycle and the next accounting cycle. The receiver's new sub-balance is not subject to future rollovers and its valid-from date is the end date of the sub-balance that was rolled over.
6. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER generates the **/event/billing/cycle/rollover_transfer** event, which has the same GLIDs as the balance impacts of the original rollover event.
7. If delayed billing is configured and rolled-over balances were consumed by the sender by delayed events, PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER receives the rollover correction event and adjusts the rollover transfer amount of the receiver accordingly.

> **Note:**
>
> You must perform rerating for the receiver if events rated before the rollover correction were consumed from the rollover transfer amount.

5

Activity Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) activity opcode workflows.

For information about the activity opcodes, see "Activity FM Policy Opcodes" and "Activity FM Standard Opcodes".

Topics in this document:

- Opcodes Described in This Chapter
- Managing Sessions
- Customizing News Feed Values and Events
- Managing Operation Failure Records

Opcodes Described in This Chapter

Table 5-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 5-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_END_SESSION	Recording Session Events Recording the End of a Session
PCM_OP_ACT_FIND_VERIFY	Starting Sessions
PCM_OP_ACT_LOAD_SESSION	Loading Sessions
PCM_OP_ACT_LOGIN	Starting Sessions
PCM_OP_ACT_LOGOUT	Recording the End of a Session
PCM_OP_ACT_POL_PROCESS_EVENTS	Customizing News Feed Values and Events
PCM_OP_ACT_POL_REQUEST_CREATE	Recording Failed Operations
PCM_OP_ACT_REQUEST_CREATE	Recording Failed Operations
PCM_OP_ACT_REQUEST_RETRIEVE	Retrieving Operation Failure Records
PCM_OP_ACT_START_SESSION	Recording the Start of a Session Starting Sessions Updating a Session Event

Table 5-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_UPDATE_SESSION	Updating a Session Event
PCM_ACT_USAGE	Managing Sessions
PCM_OP_PROCESS_EVENTS	Customizing News Feed Values and Events

Managing Sessions

BRM uses PCM_OP_ACT_USAGE to manage session events. For example, this opcode is called for a user login session.

> **Note:**
>
> This opcode is called by the other session opcodes. Do not call PCM_OP_ACT_USAGE directly.

PCM_OP_ACT_USAGE takes as input an **/account** object POID and a type-only POID that specifies the type of event being recorded.

BRM stores information about sessions in **/event/session** objects in the BRM database.

Starting Sessions

Use PCM_OP_ACT_LOGIN to start a session.

PCM_OP_ACT_LOGIN performs these operations:

1. Calls PCM_OP_ACT_FIND_VERIFY to verify the identity of the account.

   PCM_OP_ACT_FIND_VERIFY also checks whether the account specified during login is locked and whether the password entered is valid. If the password entered is not valid, it does the following:

   - Increments PIN_FLD_LOGIN_ATTEMPTS.
   - Locks the account if PIN_FLD_LOGIN_ATTEMPTS equals the value specified in the **MaxLoginAttempts** entry in the **bus_params_act.xml** file.

   These checks are applicable only to **/service/pcm_client** and **/service/admin_client**.
2. Calls PCM_OP_ACT_START_SESSION to start a login session.
3. Returns the user's **/account** POID.

PCM_OP_ACT_LOGIN is usually called by PCM_CONNECT, but it is also called directly. If called by PCM_CONNECT, this opcode ignores the database number in the application's **pin.conf** file and searches all schemas for the account.

PCM_OP_ACT_LOGIN also stores the PIN_FLD_NAP_IP_ADDRESS in the **/event/session** object on successful login. This information is used for logging CSR activities.

Recording the Start of a Session

Use PCM_OP_ACT_START_SESSION to record the start of a session event. This opcode records any type of session event object including details specific to the event type.

Sessions can be started for either **/account** or **/service** objects. The **/account** object must be specified for both cases.

- When the **/account** object is used alone, PCM_OP_ACT_START_SESSION starts a session for the **/account** object.
- When both **/account** and **/service** object POIDs are specified, PCM_OP_ACT_START_SESSION starts a session for the **/service** object.
- When a **NULL /service** object is specified, PCM_OP_ACT_START_SESSION starts a session for the **/account** object.

Session details are recorded in the base **/event/session** object. You can create an inherited class from the **/event/session** object and add fields to describe a specific type of session in more detail.

When a session of the specified type occurs, the input flist to PCM_OP_ACT_START_SESSION can specify the type of event object to create with the PIN_OBJ_TYPE field and include the detailed information fields in the PIN_FLD_INHERITED_INFO substructure. This enables any amount of detail to be recorded for any number of session types.

In the PIN_FLD_INHERITED_INFO array, the program supplies a new storable class definition. You determine the fields necessary in the new inherited storable class type and define them by using PIN_MAKE_FLD. This procedure is explained in the **pcm.h** file. PCM_OP_ACT_START_SESSION then automatically creates storable class definitions and supplies updated release files containing the new storable class definitions.

BRM controls how PCM_OP_ACT_START_SESSION records the start of a session event by using these flags:

- **PCM_OPFLG_READ_RESULT**

  When this flag is set, PCM_OP_ACT_START_SESSION returns all fields in the event object in addition to the **/account** POID.

- **PCM_OPFLG_CALC_ONLY**
  - When this flag is set, PCM_OP_ACT_START_SESSION returns the fields that would have been used to create the event object. No fields in the database are changed, and the event object is not actually created.
  - When the flag is *not* set, PCM_OP_ACT_START_SESSION creates an **/event/session** object to record the details of the session event.

Updating a Session Event

Use PCM_OP_ACT_UPDATE_SESSION to update information in a session event.

> **Note:**
>
> The session event object must already have been created by a call to PCM_OP_ACT_START_SESSION or by an opcode that calls that opcode.

PCM_OP_ACT_UPDATE_SESSION updates the PIN_FLD_DESCR field and any inherited data fields in the session object. Only the fields specified in the opcode's input flist are updated; all other fields in the **/event/session** object are left unchanged. This opcode uses the generalized PIN_FLD_INHERITED_INFO substruct so it can update **/event/session** objects of any type.

At the end of sessions, PCM_OP_ACT_UPDATE_SESSION records the session end time.

Recording the End of a Session

Use PCM_OP_ACT_LOGOUT to record the end of a login session. It calls PCM_OP_ACT_END_SESSION to end the session and assess any changes.

Recording Session Events

Use PCM_OP_ACT_END_SESSION to record the end of a session event. PCM_OP_ACT_END_SESSION performs these operations:

1. Records the session's ending timestamp.
2. Updates the PIN_FLD_DESCR field and any inherited data fields in the **/event/session** object.
3. Returns the following, depending on the flags passed in:
   - When the PCM_OPFLG_READ_RESULT flag is set, this opcode returns all fields in the event object, in addition to the POID.
   - When the PCM_OPFLG_CALC_ONLY flag is set, this opcode returns the fields that would have been used to create the event object. No fields in the database are changed, and the event object is not actually created.
   - When no flags are set, this opcode returns the POID of the event object.

Loading Sessions

Use PCM_OP_ACT_LOAD_SESSION to create and record a session event as a single operation. This opcode is valuable as a tool to load sessions in real time.

Customizing News Feed Values and Events

Use the PCM_OP_ACT_POL_PROCESS_EVENTS to process events and update the database with news feed entries. The PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode is called by the PCM_OP_PROCESS_EVENTS standard opcode.

Use the PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode to customize values in the news feed and to add new events for the news feed.

As input, this opcode takes the following information:

- PIN_FLD_EVENT: Specifies the events.

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

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

  where:

  – *numeric_IDN* is an unique identifier that maps to the reason code in the localization message. You can add multiple numeric IDs separated by **;;**.

  – *placeholder_valueN* is the value that replaces the placeholder character in the localization message for the corresponding numeric ID. The placeholder values are separated by **|~|**.

  For example:

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

  which displays the following message:

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

- PIN_FLD_FLAGS: Specifies whether to create the **/newsfeed** object or not.
  – If it is set to **1**, this opcode creates a new **/newsfeed** object.
  – If it is set to **0**, this opcode does not create a new **/newsfeed** object.

Managing Operation Failure Records

Use the following opcodes to manage operation failure records, which are stored in **/request/failed**, **/request/failed/opcode**, and **/request/failed/rest** objects:

- To create an operation failure record, use PCM_OP_ACT_REQUEST_CREATE. See "Recording Failed Operations".
- To customize an operation failure record before it is saved in the database, use PCM_OP_ACT_POL_REQUEST_CREATE. See "Recording Failed Operations".
- To retrieve one or more operation failure records, use PCM_OP_ACT_REQUEST_RETRIEVE. See "Retrieving Operation Failure Records".
- To delete an account and all operation failure records associated with it, use PCM_OP_CUST_DELETE_ACCT. See "Deleting Accounts".

To view the input and output flist specifications for these opcodes, see *BRM Opcode Flist Reference*.

Recording Failed Operations

Use the PCM_OP_ACT_REQUEST_CREATE opcode to record failed operations in a **/request/failed**, **/request/failed/opcode**, or **/request/failed/rest** object. For more information about these objects, see *Storable Class Reference*.

To record details about a failed operation, customize your client application to replace sensitive information, such as passwords, in the request payload with dummy data. Then, your client application must call the PCM_OP_ACT_REQUEST_CREATE opcode and pass the following information in the opcode's input flist:

- PIN_FLD_PARTIAL set to the status of the payload: complete (**0**) or incomplete (**1**). Incomplete indicates that sensitive information was dropped from the payload and will need to be added before the operation is reprocessed.
- PIN_FLD_STATUS set to the status of the operation: successful (**0**), failed (**1**), or written off (**2**). Written off indicates that the failed operation does not need to be reprocessed and can be deleted.
- PIN_FLD_TYPE_STR set to the MIME type of the payload, such as **text/pin_flist** for opcode flists or **application/json** for REST APIs.
- PIN_FLD_PAYLOAD_STR contains the payload data from the request in string format.
- Details about the error or failure.

PCM_OP_ACT_REQUEST_CREATE does the following:

1. Converts the payload into a buffer field.
2. Converts any error or failure details into a buffer field.
3. Calls the PCM_OP_ACT_POL_REQUEST_CREATE policy opcode to customize details about the failed operation. By default, the opcode does nothing.
4. Creates a **/request/failed**, **/request/failed/opcode**, or **/request/failed/rest** object. In multischema systems, the object is created in the same schema in which the operation failed.
5. Creates an **/event/notification/request/create** notification event.

Retrieving Operation Failure Records

Use the PCM_OP_ACT_REQUEST_RETRIEVE opcode to retrieve one or more **/request/failed**, **/request/failed/opcode**, or **/request/failed/rest** objects.

To retrieve a specific **/request** object, pass the POID of the **/request** object in the opcode's input flist.

To retrieve all **/request** objects based on specific criteria, pass the type-only POID for the **/request** object as well as the following optional input:

- To retrieve records for a specific account, pass PIN_FLD_ACCOUNT_OBJ set to the **/account** object POID.
- To retrieve records with a specific status, pass PIN_FLD_STATUS set to **0** (Successful), **1** (Failed), or **2** (Written Off).
- To retrieve records created by a specific client application, pass PIN_FLD_PROGRAM set to the name of the calling program.

6

Billing Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) billing opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Creating Bills
- Moving Bill Items
- Customizing Billing
- Managing Bill Units
- Suppressing Bills
- Managing Proforma Invoices
- Making Trial Bills
- Corrective Billing
- Using Revenue Assurance Manager

Opcodes Described in This Chapter

Table 6-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 6-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE	Customizing How to Bill Events That Occur between Billing Cycles
PCM_OP_ACT_USAGE	Customizing How to Bill Events That Occur between Billing Cycles
PCM_OP_BILL_CONVERT_AMOUNTS	About Currency Conversion
PCM_OP_BILL_CREATE_SPONSORED_ITEMS	Creating /item/sponsor Objects Running Bill Now Running Bill Now for a Service
PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS	About Currency Conversion

Table 6-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES	About Currency Conversion
PCM_OP_BILL_CYCLE_TAX	Running Bill Now
PCM_OP_BILL_MAKE_BILL	Creating Bills Modifying a Bill Object Calculating When a Bill is Due Customizing Accounting Cycles Suspending and Resuming Billing of Closed Accounts Determining Whether Bills Should Be Suppressed Managing Proforma Invoices Manually Suppressing Bills Making Trial Bills
PCM_OP_BILL_MAKE_BILL_NOW	Modifying a Bill Object Calculating When a Bill is Due Running Bill Now Running Bill Now for a Service Applying Discounts and Folds with Bill Now Creating /item/sponsor Objects Using Revenue Assurance Manager
PCM_OP_BILL_MAKE_BILL_ON_DEMAND	Using Revenue Assurance Manager
PCM_OP_BILL_MAKE_CORRECTIVE_BILL	Calculating When a Bill is Due Customizing Bill Due Date Calculations for Payment Terms Making a Corrective Bill Validating Bills for the Corrective Billing Process
PCM_OP_BILL_MAKE_TRIAL_BILL	Making Trial Bills Creating Bills Collecting Split Revenue Assurance Data
PCM_OP_BILL_MOVE_ITEM	Moving Bill Items
PCM_OP_BILL_POL_BILL_PRE_COMMIT	Modifying a Bill Object
PCM_OP_BILL_POL_CALC_PYMT_DUE_T	Creating Bills Calculating When a Bill is Due Customizing Bill Due Date Calculations for Payment Terms Changing the Bill Now Due Date Payment Due Dates for Corrective Bills
PCM_OP_BILL_POL_CHECK_SUPPRESSION	Determining Whether Bills Should Be Suppressed Adding Bill Suppression Exceptions Deleting Bill Suppression Exceptions Creating Bills
PCM_OP_BILL_POL_GET_PENDING_ITEMS	Creating /item/sponsor Objects Customizing Bill Now Running Bill Now for a Service
PCM_OP_BILL_POL_POST_BILLING	Suspending and Resuming Billing of Closed Accounts

Table 6-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_POL_SPEC_BILLNO	Customizing the Format of Bill and Invoice Numbers
PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE	Customizing Accounting Cycles
PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL	Validating Bills for the Corrective Billing Process
PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION	Ending Manual Account Suppression
PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION	Suppressing Accounts
PCM_OP_BILL_SET_BILL_SUPPRESSION	Manually Suppressing Bills
PCM_OP_BILL_SET_LIMIT_AND_CR	Customizing Credit Limit and Consumption Rules Customizing Client Applications to Modify Fixed Thresholds
PCM_OP_CUST_CREATE_BILLINFO	Creating Bill Units Updating Bill Units
PCM_OP_CUST_DELETE_BILLINFO	Deleting Bill Units
PCM_OP_CUST_POL_PREP_BILLINFO	Preparing Bill Unit Data Setting the Billing DOM According to the Payment Method Updating Bill Units Validating Bill Unit Data
PCM_OP_CUST_POL_VALID_BILLINFO	Validating Bill Unit Data Updating Bill Units Preparing Bill Unit Data
PCM_OP_CUST_SET_BILLINFO	Customizing Accounting Cycles Updating Bill Units Preparing Bill Unit Data Setting Up Hierarchical and Sharing Relationships among Bill Units Changing Hierarchical and Sharing Relationships among Bill Units
PCM_OP_CUST_SET_STATUS	Resuming Billing When Closed Accounts Are Reactivated
PCM_OP_DELIVERY_MAIL_SENDMSGS	Customizing Alert Behavior
PCM_OP_MAKE_BILL_ON_DEMAND	Modifying a Bill Object On-Purchase Billing
PCM_OP_PROCESS_AUDIT_CREATE	Using Revenue Assurance Manager
PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY	Customizing the Revenue Assurance Written-Off Event Record Summaries Using Revenue Assurance Manager
PCM_OP_PROCESS_AUDIT_POL_ALERT	Customizing Alert Behavior
PCM_OP_PROCESS_AUDIT_POL_CREATE	Customizing Audit Object Validation Using Revenue Assurance Manager
PCM_OP_PROCESS_AUDIT_SEARCH	Using Revenue Assurance Manager
PCM_OP_PYMT_COLLECT	Customizing the Minimum Amount to Charge
PCM_OP_PYMT_GRANT_INCENTIVE	Creating Bills
PCM_OP_PYMT_POL_PRE_COLLECT	Customizing the Minimum Amount to Charge
PCM_OP_SUBSCRIPTION_APPLY_RATE	Customizing Accounting Cycles

Table 6-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_CYCLE_ARREARS	Running Bill Now Customizing Accounting Cycles
PCM_OP_SUBSCRIPTION_CYCLE_FOLD	Running Bill Now
PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Running Bill Now Customizing Accounting Cycles
PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL	Customizing Accounting Cycles
PCM_OP_SUBSCRIPTION_PURCHASE_FEES	Running Bill Now
PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT	Preparing Bill Unit Data

Creating Bills

PCM_OP_BILL_MAKE_BILL creates a **/bill** object. PCM_OP_BILL_MAKE_BILL does the following:

- Applies cycle fees.
- Totals pending items in the **/bill** current total field.
- Totals open items in the **/bill** previous total field.
- If rollover correction is enabled, PCM_OP_BILL_MAKE_BILL can trigger the creation of rerating requests at the end of the delayed period if call detail records (CDRs) from the previous cycle borrow rollover from the current cycle.

  To trigger the creation of rerating requests, PCM_OP_BILL_MAKE_BILL creates a notification event of type **/event/notification/rollover_correction/rerate** for the account being billed and possibly other accounts from which a bill unit was transferred into the account.
- If rollover correction is enabled, PCM_OP_BILL_MAKE_BILL triggers rollover correction if CDRs borrow from the previous cycle borrow rollover from the current cycle.
- If payment incentives are enabled, PCM_OP_BILL_MAKE_BILL calls PCM_OP_PYMT_GRANT_INCENTIVE which determines whether to grant the payment incentive and also calculates the incentive. PCM_OP_BILL_MAKE_BILL then retrieves the payment incentives and recalculates the affected bill, changing the balance impacts accordingly.
- To determine the payment due date of a bill, PCM_OP_BILL_MAKE_BILL calls PCM_OP_BILL_POL_CALC_PYMT_DUE_T.
- At the end of the last accounting cycle in a bill unit's billing cycle, PCM_OP_BILL_MAKE_BILL calls PCM_OP_BILL_POL_CHECK_SUPPRESSION to find out whether a bill should be suppressed.

  If the PCM_OP_BILL_POL_CHECK_SUPPRESSION output flist indicates that the bill *should* be suppressed, PCM_OP_BILL_MAKE_BILL performs these tasks:

  – Generates an **/event/billing/suppression** event.

  – Extends the bill for another billing cycle instead of finalizing it.

If the PCM_OP_BILL_POL_CHECK_SUPPRESSION output flist indicates that the bill *should not* be suppressed, performs these tasks:

  - If the output flist includes an exception to bill suppression, generates an **/event/billing/suppression/exception** object and then finalizes the bill.
  - If the result does not include an exception, finalizes the bill.

  Whether or not the output flist indicates that the bill should be suppressed, PCM_OP_BILL_MAKE_BILL *always* subtracts 1 from the **/billinfo** counter that tracks the remaining cycles for which a bill has been manually suppressed (PIN_FLD_SUPPRESSION_CYCLES_LEFT) if the value in the counter is greater than **0**.

- When you use delayed billing, PCM_OP_BILL_MAKE_BILL performs some functions on the billing DOM and performs the rest of the functions at the end of the delay interval.

If the **/billinfo** object being billed is a parent, PCM_OP_BILL_MAKE_BILL creates a single **/bill** object for that parent, which includes all pending and open items from its nonpaying **/billinfo** objects.

When called by PCM_OP_BILL_MAKE_TRIAL_BILL during trial invoicing, PCM_OP_BILL_MAKE_BILL checks the value of the PIN_FLD_FLAGS field. If the value is PIN_INV_TYPE_PARENT, PCM_OP_BILL_MAKE_BILL calculates the adjusted, disputed, due, received, and writeoff values from each nonpaying child bill unit's **/invoice/trial** objects. The resulting billing totals are passed to the parent of the nonpaying child bill units.

Modifying a Bill Object

Use PCM_OP_BILL_POL_BILL_PRE_COMMIT to modify a bill object before committing it to the database. PCM_OP_BILL_POL_BILL_PRE_COMMIT is called by PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_BILL_NOW, and PCM_OP_MAKE_BILL_ON_DEMAND.

Calculating When a Bill is Due

PCM_OP_BILL_POL_CALC_PYMT_DUE_T calculates the due date and the payment collection date of a bill (**/bill** object).

PCM_OP_BILL_POL_CALC_PYMT_DUE_T is called by PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_CORRECTIVE_BILL, and PCM_OP_BILL_MAKE_BILL_NOW.

> **Note:**
>
> By default, the due date calculation is based on the time that billing is *actually* run, not on the time that a bill unit is ready to be billed.
>
> Although configurable payment collection dates are used only for BRM-initiated payment, such as payments made by credit card and direct debit, they are calculated and stored for bills associated with all payment methods.

When called by PCM_OP_BILL_MAKE_BILL or PCM_OP_BILL_MAKE_CORRECTIVE_BILL, PCM_OP_BILL_POL_CALC_PYMT_DUE_T checks the PIN_FLD_PAYMENT_TERM value in the **/payinfo** object associated with the bill's **/billinfo** object and then performs the following operations:

- If the field has no payment term ID or if the ID is **0**, PCM_OP_BILL_POL_CALC_PYMT_DUE_T uses the default process of calculating the due date rather than using the bill run management process. The default process involves the **fm_bill_pol_default_calc_due_t** function and either the PIN_FLD_DUE_DOM value or the PIN_FLD_RELATIVE_DUE_T value in the **/payinfo** object.
- If the payment term ID is *greater than* **0**, PCM_OP_BILL_POL_CALC_PYMT_DUE_T uses one of the following bill run management functions to calculate the payment due date:
  - **fm_utils_add_n_days**
  - **fm_bill_pol_add_n_bus_days**
  - **fm_bill_pol_get_nthweekdayofmonth**

  For more information, see "Customizing Bill Due Date Calculations for Payment Terms".

After calculating the due date, PCM_OP_BILL_POL_CALC_PYMT_DUE_T checks the due date adjustment value in the PIN_FLD_DUE_DATE_ADJUSTMENT field of its input flist. If the value is greater than 0, it adds the value to the due date.

PCM_OP_BILL_POL_CALC_PYMT_DUE_T then returns the payment due date to PCM_OP_BILL_MAKE_BILL, which puts it in the PIN_FLD_DUE_T field of the **/bill** object.

PCM_OP_BILL_POL_CALC_PAYMT_DUE_T does not return any values. Its output flist, however, contains the PIN_FLD_DUE_T value, which PCM_OP_BILL_MAKE_BILL and PCM_OP_BILL_MAKE_CORRECTIVE_BILL use as the due date of the bill. By default, PCM_OP_BILL_POL_CALC_PAYMT_DUE_T uses this PIN_FLD_NAME input to calculate the due date based on the current time.

Customizing Bill Due Date Calculations for Payment Terms

You use payment terms to set due dates a specified number of days after the billing cycle end date or on a specified day of the month. You can customize how PCM_OP_BILL_POL_CALC_PAYMT_DUE_T calculates due dates for regular bills as well as for corrective bills.

When PCM_OP_BILL_MAKE_CORRECTIVE_BILL calls PCM_OP_BILL_POL_CALC_PAYMT_DUE_T to calculate the due date for a corrective bill, it identifies the bill from the PIN_FLD_NAME parameter. The possible values in PIN_FLD_NAME are PIN_OBJ_NAME_CORRECTIVE_BILL, PIN_OBJ_NAME_CORRECTIVE_BILL_NOW, PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND.

The value in PIN_FLD_NAME can be used to validate that PCM_OP_BILL_POL_CALC_PAYMT_DUE_T is calculating the due date for a corrective bill and, if necessary, provide custom logic to calculate due dates for the corrective bill.

For each payment term in your system, you must customize PCM_OP_BILL_POL_CALC_PYMT_DUE_T to specify which function and parameters to use to calculate the due dates of the bills associated with the payment term. To customize the function:

1. In the function's **switch** statement, add a **case** for each payment term defined in your **/config/payment_term** object.

   The ID (constant expression) of each case should correspond to the ID of a payment term element in the PIN_FLD_PAYMENT_TERMS array of **/config/payment_term**. For example, if the array contains payment term element **1001**, add **case 1001** to the statement.

> **Note:**
>
> Case numbers **1** through **1000** are for BRM use only.

2. In each **case**, call the appropriate function with the appropriate parameters.

To calculate payment due dates for bills associated with a payment term, call one of these functions in the corresponding **case** in PCM_OP_BILL_POL_CALC_PYMT_DUE_T:

- For payment terms that add *any type of day* (business, weekend, holiday, and so on) to the billing cycle end date, use this function:

  fm_utils_add_n_days(*n*, &due_t)

  – *n* specifies the number of days to add to the billing cycle end date. The value can be any positive integer.

  – **&due_t** is a pointer to the end date of the current billing cycle (PIN_FLD_ACTG_LAST_T in the **/billinfo** object). *n* is added to this date.

  For an example, see **case TERM1** in the PCM_OP_BILL_POL_CALC_PYMT_DUE_T code sample.

- For payment terms that add *only business days* to the billing cycle end date, use this function:

  fm_bill_pol_add_n_bus_days(ctxp, n**, "default", &due_t, ebufp)**

  For an example, see **case TERM2** in the PCM_OP_BILL_POL_CALC_PYMT_DUE_T code sample.

  – **ctxp** points to a communication channel between the client application and the database.

  – n specifies the number of *business days* to add to the billing cycle end date. The value can be any positive integer.

  – **default** is the case-sensitive name of the default billing calendar.

  To use a different calendar, replace **default** with the value in the PIN_FLD_NAME field of the **/config/calendar** object that you want to use.

  > **Note:**
  >
  > * This function automatically skips weekends, so you do not need to include Sundays and Saturdays in billing calendars used only by this function.
  > * If the Connection Manager (CM) cache does not contain the specified **/config/calendar** object, the function uses the default process of calculating the payment due date rather than the bill run management process.

  – **&due_t** is a pointer to the end date of the current billing cycle (PIN_FLD_ACTG_LAST_T in the **/billinfo** object). n is added to this date.

  – **ebufp** is a pointer to the error buffer that contains any errors that occur when the function tries to retrieve the specified calendar from the database.

For an example, see **case TERM1** in the PCM_OP_BILL_POL_CALC_PYMT_DUE_T code sample.

- For payment terms that specify *a particular day of month*, use this function:

  fm_bill_pol_get_nthweekdayofmonth(d, n, &due_t)

  - **d** is a day of the week. The value can be **0** (Sunday), **1** (Monday), **2** (Tuesday), **3** (Wednesday), **4** (Thursday), **5** (Friday), or **6** (Saturday).
  - **n** is the ordinal rank of d in a month. The value can be **1** (first d of the month), **2** (second d of the month), **3** (third d of the month), or **4** (fourth d of the month).
  - **&due_t** is a pointer to the end date of the current billing cycle (PIN_FLD_ACTG_LAST_T in the **/billinfo** object).

  > **Note:**
  >
  > If **&due_t** is *after* **d n** of the month, **d n** of the *next* month is used.
  >
  > For example, if **d** = 2, **n** = 3, and **&due_t** = April 19, 2004, the payment due date is April 20, 2004 (third Tuesday of April).
  >
  > But, if **d** = 2, **n** = 3, and **&due_t** = April 21, 2004, the payment due date is May 18, 2004 (third Tuesday of the next month).

  For an example, see **case TERM3** in the PCM_OP_BILL_POL_CALC_PYMT_DUE_T code sample.

- You can also create custom functions to calculate payment due dates:

  *custom_function***(***custom_parameter***, &due_t, ebufp)**

  - **&due_t** is a pointer to the end date of the current billing cycle (PIN_FLD_ACTG_LAST_T in the **/billinfo** object). The function must take **due_t** as a parameter.
  - **ebufp** is a pointer to the error buffer that contains any errors that occur when the function tries to retrieve an object from the database. If the function uses BRM objects, this parameter is required.

The following sample is the default implementation of the **fm_bill_pol_calc_pymt_due_t** function in the **fm_bill_pol_calc_pymt_due_t.c** file:

```
switch (ptt)
    {
        case TERM1: /* Add 17 days to the billing cycle end date */
            fm_utils_add_n_days(17, &due_t);
            break;

        case TERM2: /* Add 14 business days to the billing cycle end date */
            fm_bill_pol_add_n_bus_days(ctxp, 14, "default", &due_t, ebufp);
            if (PIN_ERR_IS_ERR(ebufp))
            {
                PIN_ERR_LOG_EBUF(PIN_ERR_LEVEL_ERROR,
                "fm_bill_pol_calc_pymt_due_t: Error while using Payment Term 2", ebufp);
                goto cleanup;
            }
            /*********************************************************
            * If due_t is not changed, it means that there
            * are no /config/calendar objects available in the CM cache.
            * In this case, use the default TERM0 implementation.
```

```
            ***********************************************************/
            if (due_t == *eff_tp)
            {
                PIN_ERR_LOG_MSG(PIN_ERR_LEVEL_WARNING, "Switching to default
calculation");
                fm_bill_pol_default_calc_due_t(&due_t, eff_tp, r_due_t, due_dom, ebufp);
            }

            break;

        case TERM3: /* Make the due date the 3rd Tuesday of the month */
            fm_bill_pol_get_nthweekdayofmonth( 2, 3, &due_t);
            break;

        /* Add implementation for handling more payment terms here */

        default: /* TERM0 for backward compatibility */
            fm_bill_pol_default_calc_due_t(&due_t, eff_tp, r_due_t, due_dom, ebufp);
    }
```

Customizing the Format of Bill and Invoice Numbers

By default, bill numbers for regular bills are **B1-1**, **B1-2**, **B1-3**, and so on. For corrective bills, the default numbering is **CB1-1**, **CB1-2**, **CB1-3**, and so on.

The default numbering takes the format *Prefix*DB_num-*Seq_Number*, where:

- *Prefix* represents the prefix to be used for the bill.

  For regular bills, BRM assigns **B***DB_num* as the default prefix. *DB_num* indicates the number of the database in which the original bill is stored. For example, when *DB_num* is 1, 3, or 7, the prefixes for regular bills are **B**1, **B**3, or **B**7.

  For corrective bills, BRM assigns **CB***DB_num* as the default prefix. For example, when *DB_num* is 1, 3, or 7, the prefixes for corrective bills are **CB**1, **CB**3, or **CB**7.

- *Seq_Number* is the unique sequence number from the appropriate **/data/sequence** object. BRM supports two types of **/data/sequence** objects, (PIN_SEQUENCE_TYPE_BILL_NO and PIN_SEQUENCE_TYPE_CORR_BILL_NO).

  > **Note:**
  >
  > Of these two **/data/sequence** objects, PIN_SEQUENCE_TYPE_CORR_BILL_NO is used exclusively for corrective bills.

  For regular bills, BRM uses the unique sequence number from the PIN_SEQUENCE_TYPE_BILL_NO **/data/sequence** object.

  For corrective bills, you can select to use the unique sequence number from either of the two **/data/sequence** objects, (PIN_SEQUENCE_TYPE_BILL_NO and PIN_SEQUENCE_TYPE_CORR_BILL_NO).

A regular bill and all its corrective bills have the same POID (the bill object identifier in the BRM database). As a result, you can use the bill POID to retrieve the complete set of bills, that is, the corrected bill and its prior bills.

You can customize the prefix and the numbering sequence for bills. Use PCM_OP_BILL_POL_SPEC_BILLNO to customize bill numbers. By default, if the bill number

is in the input flist, the opcode returns it. Otherwise, the opcode generates a bill number based on the bill POID.

To use a different bill number format, use PCM_OP_WRITE_FLDS to modify the PIN_FLD_HEADER_STR field in the **/data/sequence** object. For example, to use a bill number format with numbers only and no letters, such as 100, 101, 102, and so on, set PIN_FLD_HEADER_STR to two colons (**::**).

PCM_OP_BILL_POL_SPEC_BILLNO assigns a default number to the **/account** object in the database. This opcode is called by the **pin_fld_billno** utility and uses the default implementation information to create a unique billing number. The billing number is then returned to the object in the database.

PCM_OP_BILL_POL_SPEC_BILLNO is called by PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_BILL_NOW, and PCM_OP_MAKE_BILL_ON_DEMAND.

Running Bill Now

Bill now runs PCM_OP_BILL_MAKE_BILL_NOW on a specified bill unit (**/billinfo** object) immediately. If a bill unit is not specified, this opcode creates one **/bill** for each **/billinfo** for the given account.

The PIN_FLD_NAME field in the **/bill** object contains the type of billing the **/bill** object is for:

- Regular billing
- Billing on purchase
- Bill now
- Bill now for the current cycle
- Bill now on the next cycle

The last two options enable the creation of two bills during the delayed period if your customer management system (CMS) supports that functionality. One bill is generated for the current cycle charges; the other is generated for the next cycle charges.

PCM_OP_BILL_MAKE_BILL_NOW performs the following tasks:

- Applies cycle fees, including deferred fees and folds, by calling the following opcodes:
  - PCM_OP_SUBSCRIPTION_PURCHASE_FEES
  - PCM_OP_SUBSCRIPTION_CYCLE_ARREARS
  - PCM_OP_SUBSCRIPTION_CYCLE_FOLD
  - PCM_OP_SUBSCRIPTION_CYCLE_FORWARD
- If called for a service of a sponsored account, PCM_OP_BILL_MAKE_BILL_NOW calls PCM_OP_BILL_CREATE_SPONSORED_ITEMS The bill produced is for the pending items for the specified service of the sponsored account.
- If called on a nonpaying bill unit, creates one bill for the parent bill unit that includes only the items from the nonpaying bill unit. In such cases, the PIN_FLD_GROUP_OBJ field in the **/event/billing/cycle/tax** object contains the POID of the nonpaying **/billinfo** object. If called on a parent bill unit, creates one bill that contains a total of the items from both the parent and any nonpaying bill units. This includes any nonpaying bill unit cycle taxes. In such cases, the PIN_FLD_GROUP_OBJ field contains the POID of the parent **/billinfo** object.
- Finalizes the bill.

- If configured, calls PCM_OP_BILL_CYCLE_TAX to calculate taxes.

> **Note:**
>
> Bill Now ignores the **CycleTaxInterval** business parameter and always rolls activities for each nonpaying bill unit into the parent bill unit and calculates taxes for the parent only.

- If configured, prorates cycle arrears and cycle forward arrears fees.

Bill Now does not generate invoices. You must separately run the **pin_inv_accts** utility or the **pin_bill_day** script.

Customizing Bill Now

By default, Bill Now generates a bill that includes all pending items. You can customize Bill Now to include only specified pending items. To change the default behavior, edit the search criteria in PCM_OP_BILL_POL_GET_PENDING_ITEMS.

If the bill is produced for the parent bill unit, this bill, by default, includes pending items from the parent and all nonpaying child bill units. To include items for just one of the nonpaying bill units, add functionally to PCM_OP_BILL_POL_GET_PENDING_ITEMS to filter out the rest of the items for the bill units associated with the parent bill unit.

Running Bill Now for a Service

You can extend your customer management application to generate a Bill Now type of bill for a specific service. When the selected service belongs to a member account in a charge or discount sharing group, a bill can be generated for the owner account of the sharing group.

Use the following opcodes:

- PCM_OP_BILL_MAKE_BILL_NOW
- PCM_OP_BILL_CREATE_SPONSORED_ITEMS
- PCM_OP_BILL_POL_GET_PENDING_ITEMS

Applying Discounts and Folds with Bill Now

To apply discounts or folds, your customer management application needs one of the values listed in Table 6-2 in the PIN_FLD_FLAGS field in the PCM_OP_BILL_MAKE_BILL_NOW input flist:

Table 6-2 Values to Apply Folds, Discounts, or Both

To Include:	Set the Value To:
Folds	**16**
Discounts	**32**
Folds *and* discounts	**48** (Add the values to get both actions.)

Note:

- A billing-time discount is not allowed when Bill Now includes only selected items or a specific service.
- When the billing-time discount flag is specified, you must ensure that a fold charge is configured for the specific balance used by the billing-time discount. In addition to the billing-time discount flag, you must also specify the fold flag to fold the balance used by the discount. If a fold charge is not configured for the balance used by the discount or if the fold flag is not specified, the billing-time discount is applied again during regular billing.
- When the fold flag is specified, all balances are folded, even those not used by the billing-time discount. For example, when closing an account, you can specify the fold flag to fold all the balances. In other cases, you might not want to fold all the balances. In such cases, do not specify the billing-time discount or fold flag if there are other balances that are not used by the billing-time discount.

Changing the Bill Now Due Date

The default due date for a bill created with Bill Now is calculated as the billing cycle length minus one day after Bill Now is run:

date_of_bill + billing_cycle_length - one_day

For example, if you run Bill Now on June 2, and the billing cycle is one month, the bill is due July 1.

To change the Bill Now due date to, for example, date_of_bill + billing_cycle_length - 7 days, you customize the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode. See "Customizing Bill Due Date Calculations for Payment Terms"

On-Purchase Billing

PCM_OP_BILL_MAKE_BILL_ON_DEMAND creates a **/bill** object immediately when a bundle or package that is flagged for on-purchase billing is purchased.

To create a bill on purchase for a bundle or package, the PCM_FLD_ON_DEMAND_INFO field must be set in the **/deal** or **/plan** objects.

Use PDC to set this field by selecting **Bill on Purchase** in the General Information section of the bundle or package. Use Pricing Center to set this field by selecting **Bill on Demand** on the plan or deal **Attributes** tab.

Creating /item/sponsor Objects

Use PCM_OP_BILL_CREATE_SPONSORED_ITEMS to create **/item/sponsor** objects for sponsoring accounts. These item objects include charges from the sponsored accounts. PCM_OP_BILL_CREATE_SPONSORED_ITEMS sends that information to PCM_OP_BILL_POL_GET_PENDING_ITEMS.

PCM_OP_BILL_CREATE_SPONSORED_ITEMS is called by PCM_OP_BILL_MAKE_BILL_NOW when it creates a bill for a sponsoring account. PCM_OP_BILL_CREATE_SPONSORED_ITEMS can also be run separately. In this case, the

returned list of items can be passed to PCM_OP_BILL_MAKE_BILL_NOW to produce a bill for each sponsor account.

If PCM_OP_BILL_CREATE_SPONSORED_ITEMS is called, billing time discounts and folds are not applied.

Customizing Billing

See the following topics:

- Customizing the Minimum Amount to Charge
- Customizing Accounting Cycles
- Customizing How to Bill Events That Occur between Billing Cycles
- Customizing Credit Limit and Consumption Rules
- About Currency Conversion

Customizing the Minimum Amount to Charge

You set the default minimum amount to charge a customer in the CM **pin.conf** file **minimum** entry. To check a batch of charges and refunds for any amounts below the minimum before charging and refunding customers, use PCM_OP_PYMT_POL_PRE_COLLECT. The PIN_FLD_SESSION_OBJ field in the input flist references the type of session in which the event occurred: either **/event/billing/batch/refund** or **/event/billing/batch/payment**, depending on the batch type.

> **Note:**
>
> Ensure that the minimum credit card charge does not conflict with the minimum amount to collect.

You can change the minimum credit card charge amount by modifying the default minimum payment amount in PCM_OP_PYMT_POL_PRE_COLLECT.

Before performing the charges and refunds, PCM_OP_PYMT_COLLECT allocates the PIN_FLD_CHARGE array elements to open items and then calls PCM_OP_PYMT_POL_PRE_COLLECT.

PCM_OP_PYMT_POL_PRE_COLLECT then checks each element of the input PIN_FLD_CHARGES array to ensure:

- The result of selecting open items for allocating charges is set to PIN_SELECT_RESULT_PASS.
- The amount charged is greater than or equal to the minimum payment amount.
- The amount refunded is greater than or equal to the minimum and the account has a negative balance.
- The value of the input PIN_FLD_COMMAND field is valid.

By default, the results can be the following:

- If the amount charged is less than the minimum amount, PCM_OP_PYMT_POL_PRE_COLLECT sets the PIN_FLD_DESCR field to “Below minimum" and the result to PIN_CHARGE_RES_FAIL_NO_MIN.
- If the amount refunded is less than the minimum amount, PCM_OP_PYMT_POL_PRE_COLLECT sets the PIN_FLD_DESCR field to “Below minimum" and the result to PIN_CHARGE_RES_FAIL_NO_MIN.
- If PIN_FLD_COMMAND is set to PIN_CHARGE_CMD_REFUND and the account balance is zero or higher, PCM_OP_PYMT_POL_PRE_COLLECT sets the PIN_FLD_DESCR field to “No credit available" and the result to PIN_CHARGE_RES_NO_CREDIT_BALANCE.

You can change the minimum credit card charge amount by modifying the default minimum payment amount in PCM_OP_PYMT_POL_PRE_COLLECT.

You can also customize PCM_OP_PYMT_POL_PRE_COLLECT to retrieve soft descriptor information that enables you to display the name under which you do business (your DBA name), charge offer name, and customer service number on your customer's checking account or credit card statement. See "Adding Soft Descriptors to Invoices".

Customizing Accounting Cycles

To customize accounting cycles, use PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE. This opcode can be modified to calculate the next and future accounting cycles appropriate for your business policy.

PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE is called from PCM_OP_BILL_MAKE_BILL or PCM_OP_CUST_SET_BILLINFO whenever BRM calculates PIN_FLD_ACTG_NEXT_T and PIN_FLD_ACTG_FUTURE_T.

By default, PIN_FLD_ACTG_NEXT_T is calculated if you do not specify it in the input flist, but PIN_FLD_ACTG_FUTURE_T will always be calculated based on PIN_FLD_ACTG_NEXT_T.

PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE can be modified to calculate the next and future accounting cycles appropriate for your business policy.

To customize the time interval for applying cycle forward and cycle arrears fees for a specified charge offer, use PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL.

PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL is called by PCM_OP_SUBSCRIPTION_APPLY_RATE, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD, and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS.

PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL is an empty hook. If you customize this policy opcode to pass a PIN_FLD_FLAGS output flist field set to **1**, the charge offer will be forced into a long cycle. This adds any extra days to the next accounting cycle.

Customizing How to Bill Events That Occur between Billing Cycles

Use the PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE policy to specify in which billing cycle to apply an event when an event occurs between the end of a billing cycle and when billing applications are run.

By default, this opcode selects the current month's bill, but you can customize this opcode to select the previous month's bill.

You specify how long after the billing cycle ends that new events are included in the previous month's bill by using the **ConfigBillingCycle** business parameter in the **billing** instance of the **/config/business_params** object. See "Billing Cycle Override for Delayed Billing" in *BRM Configuring and Running Billing*.

PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE is called by PCM_OP_ACT_USAGE when both of the following are true:

- The value of the **ConfigBillingCycle** business parameter is greater than **0**.
- The value of **ConfigBillingCycle** is less than or equal to the value of the **ConfigBillingDelay** business parameter.

  If **ConfigBillingCycle** is greater than **ConfigBillingDelay**, the CM returns an error.

You can customize PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE to point qualifying events to either the previous month's bill or the current month's bill.

Customizing Credit Limit and Consumption Rules

PCM_OP_BILL_SET_LIMIT_AND_CR sets the credit limit and consumption rules for both currency and noncurrency sub-balances.

By default, PCM_OP_BILL_SET_LIMIT_AND_CR sets or changes the credit limit and consumption rules in the account's default **/balance_group** object. To set credit limit and consumption rules for any of the other billing entities associated with the object, specify them with the optional PIN_FLD_BAL_GRP_OBJ field passed in the input flist.

PCM_OP_BILL_SET_LIMIT_AND_CR includes the following flags:

- If the PCM_OPFLG_READ_RESULT flag is set, all fields in the event object are returned in addition to the POID.
- If the PCM_OPFLG_CALC_ONLY flag is set, no fields in the database are changed and the event object is not actually created. The fields that would have been used to create the event object are returned to the caller.
- If the PCM_OPFLG_CALC_ONLY flag is not set, the **/event/billing/limit** object is created to record the details of the operation.

Customizing Client Applications to Modify Fixed Thresholds

To modify any fixed threshold values in a customer's credit limit data, you must customize your client application to call the PCM_OP_BILL_SET_LIMIT_AND_CR opcode.

You pass the fixed threshold values in the input flist's PIN_FLD_THRESHOLDS array. Each fixed value has a separate PIN_FLD_THRESHOLD flist entry in the array.

When the opcode receives fixed threshold values in its input flist, PCM_OP_BILL_SET_LIMIT_AND_CR updates the amount, refreshes the cache of credit limit data, and generates a **ModifyBalanceGroup** business event for ECE.

About Currency Conversion

Currency conversion is performed by the following opcodes:

- PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS converts the currency according to the conversion rate defined in the **/config/currency/conversionrates** object. You can set multiple time periods for conversion rates.

  For example, this opcode is used to convert currencies when an account using EMU currency is set up with a primary currency and a secondary currency.

  BRM supports conversion only between the euro and EMU currencies. Conversion between two EMU currencies or between any other currencies is not supported.

PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS fails if the specified time is not in the time range or if the source or destination currency is invalid. The **ERROR_NOT_FOUND** error code is returned.

- PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES retrieves the conversion rates from the **/config/currency/conversionrates** object. This opcode is called by PCM_OP_BILL_CONVERT_AMOUNTS.

  It returns the conversion rate, start and end time of the time range for this rate and currency operator information.

  PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES fails if no conversion rate is specified between the source and destination currency types. The **ERROR_NOT_FOUND** error code is returned.

Managing Bill Units

See the following topics:

- Creating Bill Units
- Updating Bill Units
- Preparing Bill Unit Data
- Validating Bill Unit Data
- Suspending and Resuming Billing of Closed Accounts
- Deleting Bill Units
- Setting Up Sharing Relationships among Bill Units
- Billing Delays for Moved Bill Units

Creating Bill Units

Use PCM_OP_CUST_CREATE_BILLINFO to create bill units.

You can set each bill unit's billing cycle fields, such as the accounting type, the billing frequency, and the billing day of month (DOM), by passing information in the opcode's input flist fields.

If the object is successfully created, the PCM_OP_CUST_CREATE_BILLINFO output flist contains:

- The POID of the **/account** object to which the bill unit belongs.
- PIN_FLD_BILLINFO array that specifies the billing information that is created.

For information about setting up relationships among bill units participating in hierarchies and charge and discount sharing groups, see "Setting Up Hierarchical and Sharing Relationships among Bill Units".

After creating a bill unit, you must associate it with the account balances for which the bill is created. To do this, the account must have multiple balance groups so that different balances can be linked to different bills. You create multiple balance groups when you create packages in PDC or plans in Pricing Center.

Setting the Billing Day of Month

A bill unit's billing day of month (DOM) is the day on which its accounting cycle and billing cycle begin.

> **Note:**
>
> The billing DOM and the accounting DOM are the same day, which is specified in the PIN_FLD_ACTG_CYCLE_DOM field of the **/billinfo** object.

To set a bill unit's billing DOM, BRM uses the following values in this order of priority:

1. **The DOM assigned to the billing segment**. BRM assigns the DOM set for the billing segment in the **/config/billing_segment** object. For more information, see "Assigning Billing DOMs Based on the Billing Segment".

   > **Note:**
   >
   > To customize how BRM assigns the DOM according to the billing segment, see "Customizing the DOM Assignment Process".

2. **The DOM used by the first bill unit in the account**. If a DOM is not assigned to the billing segment, BRM uses the DOM of the first bill unit in the account.
3. **Default setting in the CM**. If a DOM is not assigned to the billing segment and is not available from another bill unit, the DOM is set to the value assigned in the **ActgDom** business parameter.
4. **The current date**. If a DOM is not available from the billing segment, other bill units, or the CM, BRM sets the DOM to the current date.

For example, if an account was created with two bill units, **BU1** that has an assigned DOM and **BU2** that does not have a DOM assigned, BRM assigns a DOM to **BU2** as follows:

- If **BU2** is the second bill unit in the account, BRM uses the DOM of **BU1**.
- If **BU2** is the first bill unit in the account, BRM uses the DOM specified in the CM. If a value is not set in the CM, the DOM is set to the current date.

Updating Bill Units

Use PCM_OP_CUST_SET_BILLINFO to update billing information in existing bill units. This opcode calls policy opcodes to permit customization and perform validations. Set the value of the PIN_FLD_POID field in the PIN_FLD_BILLINFO array to **-1**, which causes PCM_OP_CUST_SET_BILLINFO to call PCM_OP_CUST_CREATE_BILLINFO before calling the policy opcodes.

PCM_OP_CUST_SET_BILLINFO updates an existing PIN_FLD_BILLINFO array associated with a specified account by setting new values for the array fields as specified in the input flist. Any PIN_FLD_BILLINFO array fields not included in the input flist are left unchanged.

When updating a bill unit for a large hierarchy, if the business profile for the parent account has **AsyncBDOMChange** key set to **yes** and there is an asynchronous change in the billing day of

month (DOM), the opcode uses the PIN_FLD_BILLINFO.PIN_FLD_FLAGS input flist field to determine whether to run the job for changing the billing DOM immediately or defer it for later:

- If set to immediately (**0x1**), the opcode updates the billing DOM immediately.
- If set to deferred (**0x2**), the opcode validates that the parent's business profile is set to **AsyncBDOMChange**. If the flag is set, the **/job/bdom** object is created which is later utlised by the application when it is run to update the billing DOM.

> **Note:**
>
> You must run the **pin_update_bdom** utility to process the job and update the large hierarchy's billing DOM. This is applicable only when the flag is set to deferred.

This is an optional flag and the deafult value is deferred (**0x2**).

PCM_OP_CUST_SET_BILLINFO calls PCM_OP_CUST_POL_PREP_BILLINFO to prepare the updated billing information for validation and then calls PCM_OP_CUST_POL_VALID_BILLINFO to validate the information.

Use the following opcodes to customize PCM_OP_CUST_SET_BILLINFO functionality:

- Use PCM_OP_CUST_POL_PREP_BILLINFO to prepare bill unit data. See "Preparing Bill Unit Data".
- Use PCM_OP_CUST_POL_VALID_BILLINFO to validate bill unit data. See "Validating Bill Unit Data".

Both of these opcodes are called by PCM_OP_CUST_SET_BILLINFO.

When the PCM_OP_CUST_SET_BILLINFO is called, it generates an **/event/customer/billinfo/modify** notification event that gets synchronized with the ECE.

You can set each bill unit's billing cycle fields, such as the accounting type, the billing frequency, and the billing DOM, by passing information in the opcode's input flist fields.

For information about:

- Setting up relationships among bill units participating in hierarchies and charge and discount sharing groups, see "Setting Up Hierarchical and Sharing Relationships among Bill Units".
- Changing the billing DOM for wholesale hierarchies, see "About Changing the Billing Day of Month for Wholesale Hierarchies" in *BRM Configuring and Running Billing*.

Preparing Bill Unit Data

PCM_OP_CUST_POL_PREP_BILLINFO processes the account billing fields in the **/billinfo** object during customer account creation or while updating billing information to prepare for validation. For information about the PREP opcodes, see in *BRM Developer's Guide*.

This opcode is called by PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and PCM_OP_CUST_SET_BILLINFO.

The opcode's main responsibility includes:

- Assigning default values for the billing cycle length, primary currency, and accounting type, if they are not passed in the input flist. See "Assigning Default Billing Information".

- Determining and assigning the billing DOM. See "Assigning Billing DOMs to Bill Units".

PCM_OP_CUST_POL_PREP_BILLINFO prepares the following billing information:

- Account POID
- Event ending timestamp
- Bill unit POID
- Payment method
- Bill unit name
- New or changed billing DOM
- Old billing DOM
- Accounting type: open item accounting or balance forward accounting
- Next billing date
- Billing frequency (based on the number of accounting cycles)
- Parent bill unit
- Currency used
- Secondary currency used
- Billing segment ID

PCM_OP_CUST_POL_PREP_BILLINFO provides a mechanism for taking the information passed to PCM_OP_CUST_SET_BILLINFO and processing the fields before their validation by PCM_OP_CUST_POL_VALID_BILLINFO. See "Validating Bill Unit Data".

Processing includes adding any missing fields whose values are derived or generated by PCM_OP_CUST_POL_PREP_BILLINFO, and forcing fields to predefined values independent of what you specified. You specify fields on the input flist, and this opcode returns the processed version of the data on the output flist. Validity of the field values is checked by PCM_OP_CUST_POL_VALID_BILLINFO.

If PCM_OP_CUST_POL_PREP_BILLINFO cannot derive all of the necessary fields because the values you specified are incorrect, no error is returned. Instead, the derived fields are returned on the output flist with a default value, and PCM_OP_CUST_POL_VALID_BILLINFO is called to detect the incorrect customer data and return the validation error to the calling application. This enables the calling application to get the details of the validation error instead of receiving an incorrect **ebuf** error. If PCM_OP_CUST_POL_PREP_BILLINFO cannot generate a necessary field or encounters other internal problems, it returns an **ebuf** error.

How BRM Calculates Long Billing Cycles

BRM uses the following formula to calculate long billing cycles:

Use a short cycle unless one of the following is true:

- Future billing day of month > current billing day of month

  AND

  (Future billing day of month - current billing day of month) < 15
- Future billing day of month < current billing day of month

  AND

  (Current billing day of month - future billing day of month) > 15

Examples:

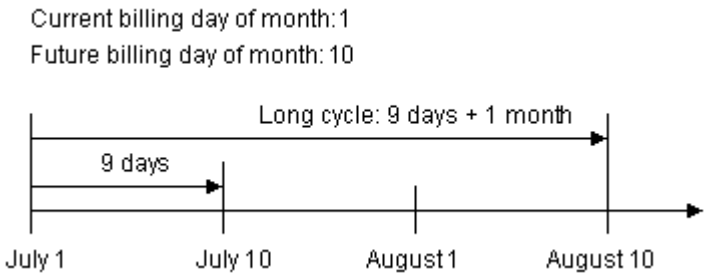
- The following example is shown in Figure 6-1. If the current billing DOM is 1 and the future billing DOM is 10:

  10 > 1

  10 - 1 = 9

  Use a long cycle.

Figure 6-1 Long Cycle Example 1



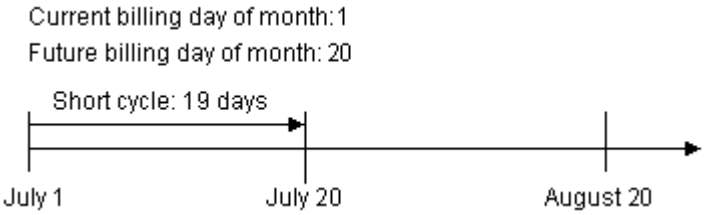
- The following example is shown in Figure 6-2. If the current billing DOM is 1 and the future billing DOM is 20:

  20 > 1

  20 - 1 = 19

  Use a short cycle.

Figure 6-2 Short Cycle Example 1



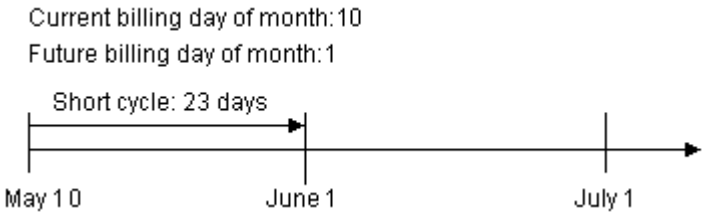
- The following example is shown in Figure 6-3. If the current billing DOM is 10 and the future billing DOM is 1:

  1 < 10

  10 - 1 = 9

  Use a short cycle.

Figure 6-3 Short Cycle Example 2



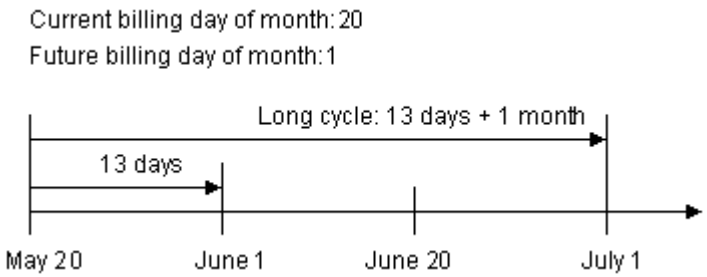
- The following example is shown in Figure 6-4. If the current billing DOM is 20 and the future billing DOM is 1:

  1 < 20

  20 - 1 = 19

  Use a long cycle.

Figure 6-4 Long Cycle Example 2



Setting the Billing DOM According to the Payment Method

You can set the billing DOM for new customers according to the payment method. For example, you can set up all accounts that pay for bills using the invoice payment method to be billed for those bills on the same day. To do this, customize the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode. Also, use event notification to implement your customization when existing customers change payment methods.

Assigning Default Billing Information

PCM_OP_CUST_POL_PREP_BILLINFO prepares the billing information for online account creation by implementing the following default values:

- If not already specified in the input flist, the PIN_FLD_BILL_WHEN field is set to the value specified in the **BillWhen** business parameter. The default value of this parameter is **1**.
- If not already specified in the input flist, the PIN_FLD_CURRENCY field is set to the value specified in the CM **pin.conf** file's **currency** entry. If no value is specified, the field is set to the currency associated with the system account.
- If not already specified in the input flist, the PIN_FLD_ACTG_TYPE field is set to the value specified in the **ActgType** business parameter. The default value of this parameter is **balance_forward**.
- If the primary currency is EURO, the PIN_FLD_CURRENCY_SECONDARY field is set to the default secondary currency.

Determining the Accounting Type

BRM determines a bill unit's accounting type by reading and using the following accounting type settings in this order:

1. **Client application or opcode flist setting**. You can specify a bill unit's accounting type when you create or modify an account in Billing Care or Customer Center.

   If you use a custom client application, you can specify a bill unit's accounting type by passing it in the PIN_FLD_ACTG_TYPE input flist field of the following opcodes:

- PCM_OP_CUST_COMMIT_CUSTOMER
- PCM_OP_CUST_MODIFY_CUSTOMER
- PCM_OP_CUST_UPDATE_CUSTOMER

2. **Default setting in the Connection Manager (CM)**. You can specify a default accounting type by using the **ActgType** business parameter. BRM uses the default setting during account creation only if an accounting type is not passed in the input flist.
3. **System-wide accounting type**. If an accounting type is not passed in the input flist nor set in the CM **pin.conf** file, BRM automatically sets the bill unit's accounting type to balance forward accounting.

Assigning Billing DOMs to Bill Units

PCM_OP_CUST_POL_PREP_BILLINFO assigns a billing DOM to a bill unit based on the following order of priority:

1. **The DOM assigned to the billing segment.** The opcode assigns the DOM set for the billing segment in the **/config/billing_segment** object. For more information, see "Assigning Billing DOMs Based on the Billing Segment".

> **Note:**
>
> To customize how the opcode assigns the DOM according to the billing segment, see "Customizing the DOM Assignment Process".

2. **The DOM used by the other bill units in the account.** If a DOM is not assigned to the billing segment, the opcode assigns the DOM used by the other bill units in the account.
3. **Default setting in the Connection Manager.** If a DOM is neither assigned to the billing segment nor available from another bill unit, the DOM is set to the value assigned in the **ActgDom** business parameter.
4. **The current date.** If a DOM is not available from the billing segment, other bill units, or the CM, the opcode assigns the DOM to the current date passed in the PIN_FLD_END_T field of the input flist.

For example, if an account was created with two bill units, **BU1** that has an assigned DOM and **BU2** that does not have an assigned DOM, the opcode assigns a DOM to **BU2** as follows:

- If **BU2** is the second **/billinfo** object associated with the **/account** object, the opcode assigns the DOM used by **BU1**.
- If **BU2** is the first **/billinfo** object associated the **/account** object, the opcode assigns the DOM set in the CM. If a value is not set in the CM, the DOM is set to the current date passed in the PIN_FLD_END_T field of the input flist.

Assigning Billing DOMs Based on the Billing Segment

When a **/config/billing_segment** object exists in the CM cache and contains an array of billing segments, PCM_OP_CUST_POL_PREP_BILLINFO performs the following tasks:

- If the input PIN_FLD_BILLING_SEGMENT field *does not* contain a value or contains **0**, PCM_OP_CUST_POL_PREP_BILLINFO sets the output PIN_FLD_BILLING_SEGMENT field to **0**. This triggers BRM to use the default process rather than the bill cycle management process of assigning a billing DOM to the bill unit.

- If the input PIN_FLD_BILLING_SEGMENT field *does* contain a value other than **0** but the input PIN_FLD_ACTG_FUTURE_DOM field *does not* contain a value, PCM_OP_CUST_POL_PREP_BILLINFO uses the bill cycle management process of assigning a billing DOM to the bill unit:
  - By default, the opcode assigns the first open DOM in the specified billing segment, starting with the current DOM.
  - Alternatively, the opcode can use a weighted average calculation to select a billing DOM. See "Customizing the DOM Assignment Process".

> **Note:**
>
> If the input PIN_FLD_ACTG_FUTURE_DOM field *does* contain a value, that value becomes the billing DOM unless it is closed. If it is closed, PCM_OP_CUST_POL_VALID_BILLINFO returns a validation error.

The default DOM assignment process rather than the bill cycle management process is also used in these situations:

- The **/config/billing_segment** object is not in the CM cache.
- In the cached **/config/billing_segment** object, no DOMs are associated with the billing segment ID specified in the input PIN_FLD_BILLING_SEGMENT field.

Customizing the DOM Assignment Process

To increase the probability that the assigned billing DOM has the lightest billing load of all the open days in a billing segment, PCM_OP_CUST_POL_PREP_BILLINFO can optionally use a weighted average calculation to select a DOM. The calculation should factor in the total billing-run processing time of each open DOM. This information is stored in the **/config/billing_segment** object.

A sample weighted average calculation is included in the opcode file (**fm_cust_pol_prep_billinfo.c**).

To use the calculation, PCM_OP_CUST_POL_PREP_BILLINFO must call the **fm_cust_pol_prep_billinfo_get_dom_from_process_t_from_cache** function to assign a DOM to a bill unit. (By default, the opcode calls **fm_cust_pol_prep_billinfo_get_next_dom_from_cache**.)

The sample calculation works as follows:

Billing segment X, to which bill unit Y belongs, has the following open DOMs and associated billing-run processing times as shown in Table 6-3:

Table 6-3 Sample Billing DOMs

DOM/Process	Value	Value	Value	Value	Value
Open Billing DOMs	5	11	16	20	75
PIN_FLD_TOTAL_PROCESS_T (in seconds)	20000	30000	15000	45000	25000

Using the total number of seconds in two DOMs (172,800) as a constant, PCM_OP_CUST_POL_PREP_BILLINFO divides the constant by each DOM's total billing-run processing time as shown in Table 6-4:

Table 6-4 Sample Bill-Run Processing Times

DOM/Process	Value	Value	Value	Value	Value
Open Billing DOMs	5	11	16	20	27
PIN_FLD_TOTAL_PROCESS_T (in seconds)	20000	30000	15000	45000	25000
172,800/ PIN_FLD_TOTAL_PROCESS_T (results are rounded down to nearest whole number)	8	5	11	3	6

PCM_OP_CUST_POL_PREP_BILLINFO then adds the results and uses their sum as a seed value to generate a random number. In this case, the sum is 33. Assume the random number is 21.

PCM_OP_CUST_POL_PREP_BILLINFO subtracts the random number from the result in the first column of the table. If the remainder is less than 0, the opcode assigns the column's DOM to the bill unit. If the remainder is greater than 0, the opcode subtracts the remainder from the result in the next column. It continues this process until it gets a remainder that is less than 0 as shown in Table 6-5:

Table 6-5 Sample Values for Billing DOMs

DOM/Process	Value	Value	Value	Value	Value
Open Billing DOMs	5	11	16	20	27
PIN_FLD_TOTAL_PROCESS_T (in seconds)	20000	30000	15000	45000	25000
172,800/ PIN_FLD_TOTAL_PROCESS_T (results are rounded down to nearest whole number)	8	5	11	3	6
First Quotient - Random Number	21 - 8 = 13	NA	NA	NA	NA
Is result less than 0? No	13 > 0	13 - 5 = 8	NA	NA	NA
Is result less than 0? No	NA	8 > 0	8 - 11 = -3	NA	NA
Is result less than 0? Yes	NA	NA	-3 < 0	NA	NA

Based on the final result of this example calculation, PCM_OP_CUST_POL_PREP_BILLINFO sets the PIN_FLD_ACTG_FUTURE_DOM in the output flist for bill unit Y to **16**.

Validating Bill Unit Data

PCM_OP_CUST_POL_VALID_BILLINFO validates an account's billing information in the ***/billinfo*** object passed to it by PCM_OP_CUST_POL_PREP_BILLINFO during customer account creation or administrative update.

The billing information can include the following:

- Payment method
- Parent **/billinfo** object

- Next bill time
- Currency used
- Billing frequency
- Accounting cycle duration
- Accounting type
- Billing segment ID

By default, PCM_OP_CUST_POL_VALID_BILLINFO validates the PIN_FLD_ACTG_TYPE, PIN_FLD_ACCOUNT_NO, and PIN_FLD_PAY_TYPE fields according to the criteria contained in the **/config/fld_validate** object. Invalid pay types result in an error.

Valid payment methods are listed in the *BRM_home***/include/pin_cust.h** file as BILL_TYPE. Include the **pin_cust.h** file in the applications that call PCM_OP_CUST_POL_VALID_BILLINFO.

Validating Billing Segment Information

If the PIN_FLD_BILLING_SEGMENT field in the PCM_OP_CUST_POL_VALID_BILLINFO input flist contains a value other than **0**, PCM_OP_CUST_POL_VALID_BILLINFO performs the following tasks:

- It verifies that the billing segment identified in the PIN_FLD_BILLING_SEGMENT field is in the **/config/billing_segment** object.
- If the billing segment is in the **/config/billing_segment** object, PCM_OP_CUST_POL_VALID_BILLINFO checks the status of the billing DOM in the input flist PIN_FLD_ACTG_FUTURE_DOM field for the specified billing segment.

PCM_OP_CUST_POL_VALID_BILLINFO logs a validation error and returns a message to the user interface in these situations:

- The specified billing segment is not in the **/config/billing_segment** object.
- The status of the specified billing DOM is closed.

Suspending and Resuming Billing of Closed Accounts

To suspend and resume billing of a bill unit in a closed account, see the following:

- Suspending Billing of Closed Accounts
- Resuming Billing When Closed Accounts Are Reactivated

Suspending Billing of Closed Accounts

To suspend billing of a closed account, use PCM_OP_BILL_POL_POST_BILLING.

This opcode is called by PCM_OP_BILL_MAKE_BILL. It enables you to perform custom processing on a bill unit at the time of billing.

By default, PCM_OP_BILL_POL_POST_BILLING suspends billing of a specified bill unit in a closed account that has the following characteristics:

- Zero balance due for all bill units (total balance due of all open and pending items).
- For account hierarchies, the balance due amount includes the open and pending item totals of every nonpaying child bill unit.

- No billable activity since the previous bill was generated.

If the bill unit has nonpaying child bill units, they are suspended, too.

For accounts with a multimonth billing cycle, the default implementation of this opcode suspends billing at the end of the last accounting cycle.

If delayed billing is enabled, PCM_OP_BILL_POL_POST_BILLING suspends billing at the end of the billing delay interval.

To indicate billing is suspended, this opcode sets the PIN_FLD_BILLING_STATUS field in the **/billinfo** object to *inactive*.

PCM_OP_BILL_POL_POST_BILLING returns the POID of the **/billinfo** object.

Resuming Billing When Closed Accounts Are Reactivated

When an account's status is changed from *closed* to *active*, PCM_OP_CUST_SET_STATUS calls an internal opcode to resume suspended billing.

The opcode sets the PIN_FLD_BILLING_STATUS field in the account's bill units to *active* and resets the billing information. For example, it resets the last billing date (PIN_FLD_LAST_BILL_T), next billing date (PIN_FLD_NEXT_BILL_T), last accounting cycle (PIN_FLD_ACTG_LAST_T), and next accounting cycle (PIN_FLD_ACTG_NEXT_T).

> **Note:**
>
> The account status is not the same as the account's billing status. For example, an account's status can be *active* while its billing status is *inactive*.

BRM does the following:

- Resets the account bill unit billing status to *active* (PIN_BILL_ACTIVE).

  > **Note:**
  >
  > A bill unit with suspended billing has the billing status set to *inactive* (PIN_BILL_INACTIVE).

- Resets the account bill unit's next billing date (PIN_FLD_NEXT_BILL_T) based on the account's billing day of month before billing was suspended. For example, if the billing day of month was the first of every month before billing was suspended, the billing day of month is set to the first of every month when billing is resumed.
- Resets the billing start date (PIN_FLD_START_T) in the **/bill** object to the current time (when billing is resumed).

> **Note:**
>
> BRM does not automatically resume suspended billing for nonpaying bill units. To do that, you must reactivate the accounts that contain those bill units.

When you resume billing, billing is run on the next scheduled billing date. This is true even if billing would have been run when billing was suspended. That is, billing is not run immediately to accommodate billing that was missed during suspension.

Deleting Bill Units

To delete bill units, write custom code that calls PCM_OP_CUST_DELETE_BILLINFO.

If the specified bill unit is a paying parent, this opcode automatically deletes any nonpaying child bill units associated with it.

> **Note:**
>
> You cannot delete a bill unit that has pending payments.

If successful, the PCM_OP_CUST_DELETE_BILLINFO output flist contains:

- PIN_FLD_POID field set to the account POID of the **/account** object that is deleted.
- PIN_FLD_BILLINFO array that specifies the billing information that is deleted.

Setting Up Hierarchical and Sharing Relationships among Bill Units

To set up hierarchical and sharing relationships among bill units, see the following sections:

- Setting Up Hierarchical Relationships among Bill Units
- Setting Up Sharing Relationships among Bill Units

Setting Up Hierarchical Relationships among Bill Units

In account hierarchies, the bill units are the paying or nonpaying entities. To configure accounts in a hierarchy to pay their own charges or to roll their charges up to other accounts in the hierarchy, you must configure their bill units to be paying or nonpaying parents or children of one another.

> **Note:**
>
> Bill unit hierarchies can, but do not have to, match account hierarchies. Bill units in accounts that belong to different account hierarchies can form a bill unit hierarchy. For example, a bill unit belonging to an account in group A can be the nonpaying child of a paying bill unit belonging to an account in group B. See "About Bill Unit Hierarchies" in *BRM Managing Customers*.

To set up hierarchical bill unit relationships in or across accounts, write custom code that calls PCM_OP_CUST_SET_BILLINFO for each participating bill unit. Set the following fields in the input flist:

- For nonpaying bill units in an account hierarchy:
  - Set the PIN_FLD_PARENT_BILLINFO_OBJ field to specify the current bill unit's immediate parent bill unit, which can be paying or nonpaying.

  - Set the PIN_FLD_AR_BILLINFO_OBJ field to specify the paying bill unit.

    > **Note:**
    >
    > If the paying bill unit is also the immediate parent bill unit, the parent and paying bill unit fields specify the same bill unit.

  - Set the value of the PIN_FLD_PAY_TYPE field to nonpaying (PIN_PAY_TYPE_SUBORD).
- For paying bill units in an account hierarchy, set the PIN_FLD_PARENT_FLAGS field.

Setting Up Sharing Relationships among Bill Units

In charge and discount sharing groups, the bill units are the paying or nonpaying entities.

To set up sharing relationships among bill units, write custom code that calls PCM_OP_CUST_SET_BILLINFO for each participating bill unit. Set the following fields in the input flist:

- For sponsored bill units in a charge or discount sharing group, set the PIN_FLD_SPONSOREE_FLAG field.
- For sponsoring bill units in a charge or discount sharing group, set the PIN_FLD_SPONSOR_FLAG field.

Changing Hierarchical and Sharing Relationships among Bill Units

When you change an account hierarchy or a charge or discount sharing group, you might also need to update the relationships among the bill units participating in the original and modified groups. To do so, write custom code that calls PCM_OP_CUST_SET_BILLINFO, and pass the new paying/nonpaying parent/child or owner/member field settings in the input flist.

Billing Delays for Moved Bill Units

When you move a bill unit from one parent to another, the future billing date (stored in the PIN_FLD_ACTG_FUTURE_T field of the **/billinfo** object) is synchronized with the parent bill unit. However, the date that the current monthly cycle ends (stored in the PIN_FLD_ACTG_NEXT_T field) for the child bill unit is not changed to match the parent bill unit. Therefore, the first billing run following the move might be different for the parent and child bill units.

For example, a child bill unit with a billing date of the 15th is moved to a new parent bill unit that has a billing date of the 30th. If the child is moved on the 20th, it is not billed on the following 30th when the parent is billed. Instead, it is billed on the 15th of the following month. Thereafter, all billing dates are synchronized.

Suppressing Bills

The following opcodes are used to suppress bills and accounts:

- PCM_OP_BILL_POL_CHECK_SUPPRESSION. See "Determining Whether Bills Should Be Suppressed".
- PCM_OP_BILL_SET_BILL_SUPPRESSION. See "Manually Suppressing Bills".

- PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION. See "Suppressing Accounts".
- PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION. See "Ending Manual Account Suppression".
- PCM_OP_BILL_SET_BILL_SUPPRESSION. See "Manually Suppressing Bills".
- PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION. See "Suppressing Accounts".

Determining Whether Bills Should Be Suppressed

After performing the accounting activities at the end of a bill unit's billing cycle, PCM_OP_BILL_MAKE_BILL calls PCM_OP_BILL_POL_CHECK_SUPPRESSION to find out whether the bill should be finalized or suppressed.
PCM_OP_BILL_POL_CHECK_SUPPRESSION performs these tasks:

1. Checks the PIN_FLD_PAY_TYPE field in the input flist to determine whether the bill unit is nonpaying. If it is, the opcode determines that it *should not* be suppressed.
2. From the cached **/config/suppression** object, PCM_OP_BILL_POL_CHECK_SUPPRESSION gets the following information for the customer segment or segments specified in the PIN_FLD_CUSTOMER_SEGMENT_LIST field of the opcode's input flist:
   - The minimum balance required for the bill to be generated.
   - The maximum number of consecutive billing cycles for which the bill can be suppressed.

   > **Note:**
   >
   > – If the account belongs to multiple customer segments, PCM_OP_BILL_POL_CHECK_SUPPRESSION gets the lowest minimum balance and the lowest maximum cycle settings associated with the segments. The lowest settings do not have to be associated with the same customer segment.
   >
   > – If the input PIN_FLD_CUSTOMER_SEGMENT_LIST field contains a customer segment ID that is not associated with suppression information, PCM_OP_BILL_POL_CHECK_SUPPRESSION uses bill suppression information associated with the default customer segment (ID 0).
   >
   > – If the input PIN_FLD_CUSTOMER_SEGMENT_LIST field is empty, PCM_OP_BILL_POL_CHECK_SUPPRESSION uses bill suppression information associated with the default customer segment (ID 0).
   >
   > – If either of the two preceding items is true and the cached configuration object has no default customer segment *or* if the object is not in the cache, the bill does not qualify for automatic bill suppression.
3. PCM_OP_BILL_POL_CHECK_SUPPRESSION checks whether any of the following is true:
   - The amount due (PIN_FLD_TOTAL_DUE) on the bill is less than the minimum balance specified in the customer segment.

> **Note:**
>
> This check is not done on bills that do not qualify for automatic bill suppression. See step 2.

- The account is suppressed.

When an account is manually suppressed, the PIN_FLD_ACCT_SUPPRESSED field in each **/billinfo** object associated with the account is set to **1**. This value is put in the PIN_FLD_ACCT_SUPPRESSED field of the opcode's input flist.

4. The number of remaining manually suppressed billing cycles is greater than 0.

   This value comes from the PIN_FLD_SUPPRESSION_CYCLES_LEFT field of the **/billinfo** object.

5. PCM_OP_BILL_POL_CHECK_SUPPRESSION makes one of the following determinations:

   - If none of the preceding conditions is true, the bill *should not* be suppressed.
   - If at least one is true, the bill *should* be suppressed.

6. If the bill *should* be suppressed, PCM_OP_BILL_POL_CHECK_SUPPRESSION checks for any exceptions to that suppression.

   To check for exceptions, PCM_OP_BILL_POL_CHECK_SUPPRESSION gets and uses the following information:

   - Adjustment events associated with the bill's account. If an adjustment occurred after the last bill was generated, the bill must be finalized.

     > **Note:**
     >
     > In addition to adjustment events, if the *payment received* exception is in effect, PCM_OP_BILL_POL_CHECK_SUPPRESSION gets *payment* events associated with the account. In such cases, if a payment was received after the last bill was generated, the bill must be finalized. By default, this exception is commented out of the opcode. To uncomment it, see "Adding Bill Suppression Exceptions".

   - The value in the input flist PIN_FLD_NUM_SUPPRESSED_CYCLES field, which indicates for how many consecutive billing cycles, the bill has been suppressed. If the value is equal to the maximum number specified in the customer segment, the bill must be finalized.

     > **Note:**
     >
     > If the maximum number of consecutive billing cycles for which the bill can be suppressed is 0 or missing (see step 2), this exception does not apply, and the bill can be suppressed for an unlimited number of consecutive billing cycles.

   - The value in the input flist PIN_FLD_LAST_BILL_OBJ field. If NULL, it means that this is the bill unit's first bill and it must be finalized.

- The value in the PIN_FLD_STATUS field in its input flist. If the value indicates that the status of the bill's account is closed, this is the bill unit's last bill and it must be finalized.

7. PCM_OP_BILL_POL_CHECK_SUPPRESSION then makes one of the following determinations:
   - If an exception exists, the bill *cannot* be suppressed.
   - If no exception exists, the bill *can* be suppressed.

The PCM_OP_BILL_POL_CHECK_SUPPRESSION output flist contains the values listed in Table 6-6. PCM_OP_BILL_MAKE_BILL uses these values to determine whether the bill should be suppressed or finalized:

Table 6-6 Output Flist for PCM_OP_BILL_POL_CHECK_SUPPRESSION

Output Flist Field	Value	Meaning
PIN_FLD_RESULT	**0**	Bill should not be suppressed.
PIN_FLD_RESULT	**1**	Bill's balance is below the minimum required to finalize it, so bill should be automatically suppressed.
PIN_FLD_RESULT	**2**	Bill is manually suppressed.
PIN_FLD_RESULT	**3**	Account is manually suppressed.
PIN_FLD_EXCEPTION	**0**	No exception.
PIN_FLD_EXCEPTION	**1**	Payment, adjustment, or credit exception.
PIN_FLD_EXCEPTION	**2**	First bill exception.
PIN_FLD_EXCEPTION	**3**	Closed account exception.
PIN_FLD_EXCEPTION	**4**	Maximum cycle exception.

All types of bill suppression can be overridden by exceptions.

Manually Suppressing Bills

To suppress and unsuppress bills manually, use PCM_OP_BILL_SET_BILL_SUPPRESSION. This opcode performs the following tasks:

- Suppresses a bill for a specified number of billing cycles.

  If the opcode's input flist PIN_FLD_SUPPRESSION_CYCLES_LEFT field contains a value greater than 0, the opcode sets the PIN_FLD_SUPPRESSIOIN_CYCLES_LEFT field in a specified **/billinfo** object with that value.

  This value represents the number of consecutive billing cycles for which the bill is to be suppressed. At the end of each billing cycle, PCM_OP_BILL_MAKE_BILL subtracts 1 from this value in the **/billinfo** object. When the value reaches 0, bill suppression ends.

- Unsuppresses a manually suppressed bill.

  If the PIN_FLD_SUPPRESSIOIN_CYCLES_LEFT field in a **/billinfo** object contains a value greater than 0, the bill is manually suppressed for the specified number of billing cycles. To end the suppression early, call PCM_OP_BILL_SET_BILL_SUPPRESSION, specify the appropriate **/billinfo** object in the input PIN_FLD_POID field, and set the input PIN_FLD_SUPPRESSION_CYCLES_LEFT field to **0**.

- Generates an **/event/audit/suppression/bill** object each time it suppresses or unsuppresses a bill.

If you use customer segments to set the maximum number of consecutive billing cycles that bills can be suppressed, be careful not to suppress bills manually for more than the specified maximum number of cycles. When a suppressed bill exceeds the maximum consecutive cycle number, a suppression exception triggers BRM to generate the bill. This might confuse customers who requested that their bills be manually suppressed for a longer period of time.

To determine the maximum number of consecutive cycles for which a bill *can* be suppressed:

1. Find out which customer segments the bill's account belongs to by checking the PIN_FLD_CUSTOMER_SEGMENT_**LIST** field in the **/account** object.
2. Check the PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES field of the appropriate customer segment in the **/config/suppression** object.

> **Note:**
>
> - If an account belongs to more than one customer segment, the *lowest* PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES value associated with the segments applies.
> - If an account belongs to no customer segments, the PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES value of the default customer segment (ID 0) applies.
> - If an account belongs to no customer segments *and* your system has no default customer segment, the bill can be suppressed for an unlimited number of consecutive cycles.

To determine the current number of consecutive billing cycles for which a bill *has been* suppressed, check the PIN_FLD_NUM_SUPPRESSED_CYCLES field of the **/billinfo** object.

Suppressing Accounts

Use PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION to suppress accounts. Manual account suppression enables you to accomplish the following tasks:

- Suppress an account immediately or on a specified future date.

  PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION can purchase one account bundle for the account to handle any purchase and cycle fees you want to associate with account suppression. The **/deal** object POID must be passed to the PIN_FLD_DEAL_OBJ field of the opcode's input flist.

  If you use customer segments to set the maximum number of consecutive billing cycles that bills can be suppressed, be careful not to suppress accounts manually for more than the specified maximum number of cycles . When a suppressed bill exceeds the maximum consecutive cycle number, a suppression exception triggers BRM to generate the bill. This might confuse customers who requested that their bills be manually suppressed for a longer period of time.

  To determine the maximum number of consecutive cycles for which a bill can be suppressed:

  1. Find out which customer segments the bill's account belongs to by checking the PIN_FLD_CUSTOMER_SEGMENT_LIST field in the **/account** object.

2. If the account belongs to one or more customer segments, check the PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES field of the appropriate customer segments in the **/config/suppression** object.
3. If the account does not belong to any customer segments, check the value of PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES for the default customer segment (ID 0).

The maximum consecutive cycles is determined in the following way:

1. If an account belongs to more than one customer segment, the *lowest* PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES value associated with the segments applies.
2. If an account belongs to no customer segments, the PIN_FLD_MAX_SUPPRESSED_BILL_CYCLES value of the default customer segment (ID 0) applies.
3. If an account belongs to no customer segments *and* your system has no default customer segment, the bill can be suppressed for an unlimited number of consecutive cycles.

> **Note:**
>
> To determine the current number of consecutive billing cycles for which a bill has been suppressed, check the PIN_FLD_NUM_SUPPRESSED_CYCLES field of the **/billinfo** object.

- Remove the suppression from an account immediately or on a specified future date.

  If a suppression bundle was purchased when the account was manually suppressed, the bundle's POID must be passed to the FLD_DEAL_OBJ field of the opcode's input flist to cancel the bundle.

PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION performs the following tasks:

- Inactivates all services and charge offers associated with the account specified in the PIN_FLD_POID field of its input flist.
- Optionally purchases an account bundle for the account.

  The **/deal** object POID must be specified in the PIN_FLD_DEAL_OBJ field of the opcode's input flist. It is the only *active* bundle that can be associated with a suppressed account. You can use it to handle any purchase and cycle fees that you want to charge for suppressing the account.
- In each **/billinfo** object associated with the account, sets the PIN_FLD_ACCT_SUPPRESSED field to **1**.
- Generates an **/event/audit/suppression/account/on** object.

To suppress the account on a future date, set the PIN_FLD_END_T field in the PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION input flist to the appropriate date. When a date is specified in this field, the opcode schedules a call to itself on that date.

PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION does not initiate any required provisioning of deactivated services.

Ending Manual Account Suppression

PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION ends manual account suppression immediately or on a specified future date. This opcode does not initiate any required provisioning of reactivated services.

PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION performs the following tasks:

- Removes any suppression bundle associated with the account.
- Reactivates all services and charge offers associated with the account specified in the PIN_FLD_POID field of its input flist.
- In each **/billinfo** object associated with the account, sets the PIN_FLD_ACCT_SUPPRESSED field to **0**.
- Generates an **/event/audit/suppression/account/off** object.

To end account suppression on a future date, set the PIN_FLD_END_T field in the PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION input flist to the appropriate date. When a date is specified in this field, the opcode schedules a call to itself on that date.

Adding Bill Suppression Exceptions

To add a bill suppression exception to your system:

1. In the PCM_OP_BILL_POL_CHECK_SUPPRESSION source code, make the appropriate modifications to these functions:
   - **fm_bill_pol_get_suppression_reason**

     This function checks both for reasons to suppress a bill and for exceptions that override reasons to suppress.
   - **fm_bill_pol_get_payment_adjustment_event**

     If the new exception is based on an event, add the appropriate **/event** storable class to the **eventsBuf** array in this function:

```
char eventsBuf[2000] = "'/event/billing/adjustment/account',\
                    '/event/billing/refund/cash','/event/billing/refund/cc',\
                    '/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'";

                    /*********************************************************
                     * Payment exceptions are being commented/taken out of the
                     * eventBuf buffer. If required these event ids can be
                     * included into the buffer to consider the payment
                     * exceptions.
                     *********************************************************/
                    /*'/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'*/
```

> **Note:**
>
> By default, the payment-received suppression exception is commented out of the preceding code and is thus disabled. To enable it, remove the comment symbols (**/*** and ***/**) enclosing the payment events.

2. In the *BRM_home***/include/pin_bill.h** file, add the appropriate enumerated name and value to the **bill_suppression_exceptions** variable:

```
typedef enum bill_suppression_exceptions {
                    PIN_NO_EXCEPTION =                                  0,
                    PIN_DUE_TO_PAYMENT_ADJUSTMENT_MADE =                1,
                    PIN_DUE_TO_FIRST_BILL =                             2,
                    PIN_DUE_TO_ACCOUNT_CLOSED =                         3,
                    PIN_DUE_TO_MAX_ALLOWED_SUPPRESSION_COUNT_REACHED =  4
} bill_suppression_exceptions_t;
```

Do not duplicate a value, and do not delimit the last name-value pair with a comma. If you add a name-value pair to the end of the existing list, add a comma to the end of the preceding name-value pair.

The values are used to populate the PIN_FLD_EXCEPTION field of the PCM_OP_BILL_POL_CHECK_SUPPRESSION output flist.

Deleting Bill Suppression Exceptions

To delete a bill suppression exception from your system:

1. In the PCM_OP_BILL_POL_CHECK_SUPPRESSION source code, delete or comment out the appropriate code in these functions:
   - **fm_bill_pol_get_suppression_reason**

     This function checks both for reasons to suppress a bill and for exceptions that override reasons to suppress.

     > **Note:**
     >
     > Do not remove logic used by processes that check for other exceptions.

   - **fm_bill_pol_get_payment_adjustment_event**

     If the deleted exception is based on an event, remove the appropriate **/event** storable class from the **eventsBuf** array in this function:

```
char eventsBuf[2000] = "'/event/billing/adjustment/account',\
                    '/event/billing/refund/cash','/event/billing/refund/cc',\
                    '/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'";

                        /*********************************************************
                         * Payment exceptions are being commented/taken out of the
                         * eventBuf buffer. If required these event ids can be
                         * included into the buffer to consider the payment
                         * exceptions.
                         *********************************************************/
                        /*'/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'*/
```

2. In the *BRM_home*/**include/pin_bill.h** file, delete or comment out the appropriate enumerated name and value from the **bill_suppression_exceptions** variable:

```
typedef enum bill_suppression_exceptions {
                    PIN_NO_EXCEPTION =                                  0,
                    PIN_DUE_TO_PAYMENT_ADJUSTMENT_MADE =                1,
                    PIN_DUE_TO_FIRST_BILL =                             2,
                    PIN_DUE_TO_ACCOUNT_CLOSED =                         3,
                    PIN_DUE_TO_MAX_ALLOWED_SUPPRESSION_COUNT_REACHED =  4
} bill_suppression_exceptions_t;
```

If you remove the last name-value pair, also remove the comma at the end of the preceding name-value pair.

The values are used to populate the PIN_FLD_EXCEPTION field of the PCM_OP_BILL_POL_CHECK_SUPPRESSION output flist.

Managing Proforma Invoices

You call the PCM_OP_BILL_MAKE_BILL opcode to perform actions on proforma invoices. Proforma invoices can be generated by passing the appropriate flags in the PIN_FLD_FLAGS field of the input flist.

> **Note:**
>
> Proforma invoices include billing-time tax and billing-time discounts.

The bill and bill items change state and status during the proforma invoicing process. Following are the states and statuses at different stages in the process:

Before a proforma invoice is generated:

- The bill is in the state **InProgress**.
- The bill items are in the status PIN_ITEM_STATUS_PENDING.

When a proforma invoice is generated:

- The bill is in the state **PreInvoiceInProgress**.
- The bill items are in the status PIN_ITEM_STATUS_OPEN, and the state PIN_ITEM_FLAG_FROZEN is added. The items are frozen and you cannot perform A/R operations on them.

> **Note:**
>
> When a proforma invoice is generated for the first time, the following fields in the **/billinfo** object are updated to those for the next billing cycle:
>
> - Next billing date (PIN_FLD_ACTG_NEXT_T)
> - Last billing date (PIN_FLD_ACTG_LAST_T)
> - Future billing date (PIN_FLD_ACTG_FUTURE_T)

If the proforma invoice is accepted:

- The bill is in the state **New**.
- The bill items are in the status PIN_ITEM_STATUS_OPEN, and the state PIN_ITEM_FLAG_FROZEN is removed from these items.

If a proforma invoice is rejected, the status and states are changed to what they were in before the proforma invoice was generated, and the calculated totals and dues are reverted back to 0:

- The bill is in the state **InProgress**.
- The bill items are in the status PIN_ITEM_STATUS_PENDING, and the state PIN_ITEM_FLAG_FROZEN flag is removed from these items.

> **Note:**
>
> - The field PIN_FLD_PREINVOICE_VALIDITY_T under the **/bill** object indicates the validity of the proforma bill.
> - Call the PCM_OP_BILL_POL_CALC_PYMT_DUE_T opcode to override the default validity set by the **PreInvoiceValidity** business parameter.

Run invoicing with proforma invoice flags to generate proforma invoices which can be shared with the customer for verification.

The value passed in the PIN_FLD_FLAGS opcode's input flist field specifies which action is taken by the opcode: create, accept, or reject the invoice.

- To generate the proforma invoice, use the flag PIN_FLD_FLG_PRE_INVOICE (0x1000).
- To accept the proforma invoice, use the flag PIN_FLD_FLG_PRE_INVOICE_ACCEPT (0x2000).
- To reject the proforma invoice, use the flag PIN_FLD_FLG_PRE_INVOICE_REJECT (0x4000).

To regenerate the proforma invoice, you generate the proforma invoice again by adding the PIN_FLD_FLG_PRE_INVOICE flag, and include the bill object poid ID of the bill to be regenerated.

The following is an example input for regenerating a proforma invoice:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 123 0
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 123 0
0 PIN_FLD_PROGRAM_NAME   STR [0] "pin_bill_accts"
0 PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 124 5
```

```
0 PIN_FLD_BILL_OBJ       POID [0] 0.0.0.1 /bill 12345 5
0 PIN_FLD_FLAGS          INT [0] 4096
0 PIN_FLD_END_T        TSTAMP [0] (1707120000) Mon Feb  5 00:00:00 2024
```

Moving Bill Items

You can move bill items in the same bill unit from one bill to another. For example, after you have rejected a proforma invoice, you can move disputed items to another bill. The opcode PCM_OP_BILL_MOVE_ITEM moves items from one bill to another bill.

To move an item, ensure the following:

- The item is in **Pending** state.
- The **/bill** object which the item is moved from and the **/bill** object to which item is moved are in **InProgress** state.

> **Note:**
>
> If you attempt to move a bill item for which a proforma invoice has been generated or accepted, the validation fails and the item is not moved.
> To move a bill item, the bill should be in **In Progress** state. Generating a proforma invoice changes the bill state to **PreInvoiceInProgress**, and acceptance changes the bill state to **New**, causing the validation to fail.

The following is a sample input flist for PCM_OP_BILL_MOVE_ITEM:

```
# number of field entries allocated 20, used 3
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 123456 0
0 PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/usage 123456789012345678 0
0 PIN_FLD_TO_OBJ         POID [0] 0.0.0.1 /bill 123456 0
```

Making Trial Bills

When you run the **pin_trial_bill_accts** utility to perform trial billing, PCM_OP_BILL_MAKE_TRIAL_BILL is called to create trial invoices and collect revenue assurance data for the trial billing run.

> **Note:**
>
> To collect revenue assurance data for trial billing, you must enable the trial billing utility to generate revenue assurance data. See "Enabling Billing Utilities to Generate Revenue Assurance Data" in *BRM Collecting Revenue Assurance Data*.

If you enable trial billing to collect revenue assurance data, PCM_OP_BILL_MAKE_TRIAL_BILL returns the summarized data in the PIN_FLD_REVENUES_ARRAY field.

The fields on the PCM_OP_BILL_MAKE_TRIAL_BILL input flist specify if this opcode creates invoices and collects split revenue assurance data for the account specified in the input flist. If

invoices are created, this opcode returns an array of trial invoice POIDs for the invoices that were created. If split revenue assurance data is collected, this opcode returns an array of revenue amounts for each item type and associated service type. The opcode opens a separate transaction to create the trial invoices.

The PIN_FLD_PROGRAM_NAME field in the input flist should always contain **pin_trial_bill_accts** even if you call the opcode from another application.

> **Note:**
>
> If a start date is not provided, PCM_OP_BILL_MAKE_TRIAL_BILL creates trial invoices for all billing cycles that are completed before the end date and that have not been billed. For accounts with skipped billing cycles, more than one trial invoice might be created.

PCM_OP_BILL_MAKE_TRIAL_BILL calls PCM_OP_BILL_MAKE_BILL to compute the balance impacts and the balance due for the accounts specified in the input flist. The input flist includes the following fields:

- The POID of the **/account** object for trial billing.
- The name of the program that called PCM_OP_BILL_MAKE_TRIAL_BILL.
- The start and end dates that specify the billing cycles for trial billing.
- LAST_BILL_STATE_TO_PROCESS with the value **2** to indicate that the bill unit state is final.
- Two optional flag fields:
  - PIN_FLD_PREINVOICE_MODE specifies whether to generate trial invoices for the trial billing run.
  - PIN_FLD_CHECK_SPLIT_FLAGS specifies whether to split the revenue assurance data collected for trial billing into detailed categories. See "Collecting Split Revenue Assurance Data".

> **Note:**
>
> Trial billing stops and reports a warning message when it encounters an account or bill unit with inactive status.

Collecting Split Revenue Assurance Data

If the PIN_FLD_CHECK_SPLIT_FLAGS field is in the PCM_OP_BILL_MAKE_TRIAL_BILL input flist and has a value of **1**, PCM_OP_BILL_MAKE_TRIAL_BILL passes the flag to PCM_OP_BILL_MAKE_BILL, which returns amounts associated with each item type and service type combination so that revenue assurance data collected for trial billing can be split into detailed categories. The item and service type details plus the total number of subscription services associated with the account are returned in the PIN_FLD_REVENUES array in the output flist. If any bills were suppressed, the amount suppressed and the suppression reason are also returned.

If PIN_FLD_CHECK_SPLIT_FLAGS has a value of **0** or is not in the input flist, PCM_OP_BILL_MAKE_TRIAL_BILL does not return item type and service type details.

Split revenue assurance data can be viewed by generating a Revenue Assurance Billing Detail report.

Corrective Billing

See the following topics:

- Making a Corrective Bill
- Validating Bills for the Corrective Billing Process
- Payment Due Dates for Corrective Bills

Making a Corrective Bill

PCM_OP_BILL_MAKE_CORRECTIVE_BILL creates a corrective bill for a **/bill** object at the time of billing. It is called by the **pin_make_corrective_bill** utility.

If PCM_OP_BILL_MAKE_CORRECTIVE_BILL is called with the **-validate_only** parameter, the opcode does not generate a corrective bill for the selected bill, but merely validates whether a corrective bill can be generated for that bill.

The value in the PIN_FLD_FLAGS input field of the opcode determines whether the opcode merely validates the bill or actually creates the corrective bill object. The **pin_bill.h** file contains the following predefined values for these constants:

```
#define PIN_BILL_VALIDATION_ONLY 0x004
#define PIN_BILL_VALIDATION_NO_CHARGES 0x008
```

PCM_OP_BILL_MAKE_CORRECTIVE_BILL returns a value in the PIN_FLD_RESULT output field to indicate whether the bill passed or failed the validation. The **pin_bill.h** file contains the following predefined values for these constants:

```
#define PIN_BILL_VALIDATION_PASSED 0x001
#define PIN_BILL_VALIDATION_FAILED 0x002
```

PCM_OP_BILL_MAKE_CORRECTIVE_BILL does the following:

1. Locks the bill's **/billinfo** object. After BRM locks a **/billinfo** object, it does not accept any corrections until the **/billinfo** object is unlocked.

   > **Caution:**
   >
   > If User A and User B both access a bill and User A submits the bill for corrective billing, BRM locks the **/billinfo** object. BRM applies any adjustments made by User B to the corrected bill object only *after* BRM unlocks the **/billinfo** object.
   >
   > Also, if User B had applied the adjustment *before* User A submitted the bill for correction, the corrected bill will contain *both* adjustments.

2. Verifies that a corrective bill can be generated for the corrections by completing the following:
   - Standard validations for the request. See "Standard Bill Validations".
   - Policy validations. See "Policy Bill Validations". The opcode also performs any custom validations that you have added and that are necessary.

If the opcode validates that a corrective bill can be generated and such a bill is to be generated, the opcode continues its process. It creates the **/history_bills** object for the **/billinfo** object.

3. Checks to see whether the **pin_make_corrective_bills** utility called it with the **-validate_only** parameter. If so, PCM_OP_BILL_MAKE_CORRECTIVE_BILL reports whether BRM can generate a corrective bill for the selected bill and does not proceed further.
4. Assigns a new bill number for the corrective bill.
5. Sets the due date for the corrective bill.
6. Creates the **/event/billing/corrective_bill** object for the prior bill in the following way:
   - The opcode includes all the A/R actions applied or allocated to the prior bill. These actions impact the totals or balances for the respective bill items and the bill. For the bill items that do not have any A/R action, BRM stores the value from the original bill in the current bill.

     The following entries for the PIN_FLD_FLAGS field in a bill object indicate that there was an allocation of settlement or refund for the bill. The *BRM_home***/include/pin_flds.h** file contains the values that BRM uses in the PIN_FLD_FLAGS field for a corrective bill.

     ```
     #define PIN_BILL_FLG_SETTLEMENT_ALLOC 0x200
     #define PIN_BILL_FLG_REFUND_ALLOC 0x400
     ```

   - The opcode stores a correction reason you provide.
   - The opcode stores the name for the corrective bill. The *BRM_home***/include/pin_bill.h file** contains the values that BRM uses in the PIN_FLD_NAME field for a corrective bill:

     ```
     #define PIN_OBJ_NAME_CORRECTIVE_BILL "PIN Corrective Bill"
     #define PIN_OBJ_NAME_CORRECTIVE_BILL_NOW "PIN Corrective Bill Now"
     #define PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND "PIN Corrective Bill On Demand"
     ```

If PCM_OP_BILL_MAKE_CORRECTIVE_BILL encounters an error in generating the corrective bill, it logs an error against the prior bill in the **default.pinlog** utility log file and does not generate the corrective bill.

Validating Bills for the Corrective Billing Process

The corrective billing process uses PCM_OP_BILL_MAKE_CORRECTIVE_BILL to provide standard validations. This opcode calls PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL to complete the policy validations on a selected bill unit. PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL performs default policy validations and/or any custom validations that you provide. PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL validates a bill object to determine whether BRM can generate a corrective bill for it.

PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL requires values for the account POID in PIN_FLD_POID, the bill POID in PIN_FLD_BILL_OBJ, and PIN_FLD_INV_TYPE.

The PIN_FLD_FLAGS input field is optional. If present, this field indicates that there are charges in the bill to be validated by PCM_OP_BILL_MAKE_CORRECTIVE_BILL or requires the opcode to validate the A/R charges in the input bill.

PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL returns the success or failure of the validation in PIN_FLD_RESULT and the reason for the failure to validate the bill in PIN_FLD_ERROR_DESCR.

The *BRM_home*/**include**/**pin_bill.h** file contains the following values that PCM_OP_BILL_MAKE_CORRECTIVE_BILL opcode uses when it validates bills for corrective billing:

Example 6-1 Constants Associated with Validation in pin_bill.h File

```
#define PIN_BILL_VALIDATION_PASSED 0x001
#define PIN_BILL_VALIDATION_FAILED 0x002
#define PIN_BILL_VALIDATION_ONLY 0x004
#define PIN_BILL_VALIDATION_NO_CHARGES 0x008
#define PIN_BILL_VALIDATION_AR_CHARGES_EXIST 0x010
#define PIN_BILL_VALIDATION_FOR_AR_CHARGES_NEEDED 0x020
```

See the following discussions in *BRM Configuring and Running Billing*:

- Validating bills for the corrective billing process (includes the standard and policy validations BRM performs)
- **pin_make_corrective_bill** utility

Standard Bill Validations

PCM_OP_BILL_MAKE_CORRECTIVE_BILL checks to see whether you enabled corrective billing and whether the corrective bill is for a finalized bill. If the bill is part of a bill unit hierarchy, it checks whether you are requesting a corrective bill for the parent bill. The opcode returns an error if you attempt to generate a corrective bill for a nonpaying child bill in a hierarchy.

Policy Bill Validations

By default, PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL checks the bill and returns its findings to PCM_OP_BILL_MAKE_CORRECTIVE_BILL. Then, PCM_OP_BILL_MAKE_CORRECTIVE_BILL continues or terminates its process based on the output from PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL.

The checks performed by PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL are as follows:

- PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL verifies that an invoice exists for the (original or corrective) bill for which you are attempting to generate a corrective bill. If there is no invoice associated with that bill, the opcode does not permit the generation of a corrective bill. It enters an error message in the PIN_FLD_ERROR_DESCR field and stops any further validation on the bill.
- If there are no A/R charges for the prior bill, PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL verifies that it is creating only a replacement invoice for that bill. If a replacement invoice is not being generated for the corrective bill, the opcode enters an error message in the PIN_FLD_ERROR_DESCR field and stops any further validation on the bill.
- When balance forward accounting is associated with a bill, PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL verifies that the corrective bill is being generated for the last bill in the billing cycle. If this validation fails, the opcode enters a PIN_BILL_VALIDATION_BILL_NOT_LAST flag in the PIN_FLD_FLAGS field. (For open item accounting, BRM permits the generation of corrective bills for any previous cycle.)
- PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL verifies whether you enabled a corrective bill to be generated when there is a payment to be processed against that bill. If this validation fails, the opcode enters a PIN_BILL_VALIDATION_NOT_PAID flag in the PIN_FLD_FLAGS field.

- The sum of the corrections must equal or exceed the specified amount threshold necessary to create the corrective bill. If this validation fails, PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL enters a PIN_BILL_VALIDATION_AR_TOO_LOW flag in the PIN_FLD_FLAGS field.

Payment Due Dates for Corrective Bills

BRM determines due dates for regular and corrective bills by using PCM_OP_BILL_POL_CALC_PYMT_DUE_T.

By default, PCM_OP_BILL_POL_CALC_PYMT_DUE_T recalculates the due dates for corrective bills starting with the current time when the opcode generates the corrective bill and taking into account the payment terms. For example, if the payment terms set the due date for a bill to be 30 days from the billing date, and the corrective bill was generated on April 12, the due date for the corrective bill would be May 12.

Using Revenue Assurance Manager

Revenue Assurance Manager uses the following Process Audit FM standard opcodes:

- PCM_OP_PROCESS_AUDIT_CREATE

  This opcode creates audit objects for revenue assurance. It is called by PCM_OP_BILL_MAKE_BILL_NOW and PCM_OP_BILL_MAKE_BILL_ON_DEMAND.

  This opcode performs the following actions:

  - Reads the primary database ID for storing **/process_audit** objects.
  - Calls PCM_OP_PROCESS_AUDIT_POL_CREATE to validate audit data and check for duplicate audit objects.
  - Calls PCM_OP_CREATE_OBJ to create audit objects.
- PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY

  This opcode creates a summary of a write-off operation. It is triggered by a notification event generated by Suspense Manager when a suspended event record is written off. This opcode gets information about the written-off event records from the notification event, including the record's suspended batch ID and its original batch ID.
- PCM_OP_PROCESS_AUDIT_SEARCH

  This opcode searches for and returns revenue assurance data from groups of **/process_audit/batchstat** objects.

  You pass in the following information:

  - A **/process_audit/batchstat** storable subclass type.
  - A control point name.
  - The detailed grouping or summary data you want.
  - The type of search to perform.

  The opcode returns the relevant statistics.

Customizing Revenue Assurance Manager

You can use the following opcodes to customize Revenue Assurance Manager:

- PCM_OP_PROCESS_AUDIT_POL_CREATE.

See "Customizing Audit Object Validation".

- PCM_OP_PROCESS_AUDIT_POL_ALERT.

  See "Customizing Alert Behavior".

- PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY.

  See "Customizing the Revenue Assurance Written-Off Event Record Summaries".

Customizing Audit Object Validation

PCM_OP_PROCESS_AUDIT_POL_CREATE performs the following actions:

- Checks for duplicate audit objects and validates audit data.
- Performs validation only if no duplicate audit objects exist.

The opcode returns any validated data and duplicate records.

It is called by PCM_OP_PROCESS_AUDIT_CREATE

You can customize PCM_OP_PROCESS_AUDIT_POL_CREATE by modifying its flist fields, modifying duplicate checks, and adding validation checks.

Customizing Alert Behavior

Revenue Assurance Manager uses PCM_OP_PROCESS_AUDIT_POL_ALERT to send email alerts when threshold values are crossed.

PCM_OP_PROCESS_AUDIT_POL_ALERT reads the details of the alert from the notification event and sends an alert email message using the details in the **/config/ra_alerts object**. These details include the recipient email addresses, the locale to be used for the subject, and the message body text.

PCM_OP_PROCESS_AUDIT_POL_ALERT calls PCM_OP_DELIVERY_MAIL_SENDMSGS to send email messages using **dm_email**.

You can customize this opcode to notify an external system or to change the message body or subject of the email. For example, you might want to send alerts via text messages or collect alert information in error logs in addition to sending email messages.

Customizing the Revenue Assurance Written-Off Event Record Summaries

After Suspense Manager writes off an event record, Revenue Assurance Manager creates a summary of the event record by calling PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY. You can change this process by adding custom code to the opcode.

By default, PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY reads fields from **/suspended_usage/telco** objects and maps them to fields in the **/process_audit/batchstat/status** objects. Table 6-7 shows the mapping; grouped by batch ID and original batch ID:

Table 6-7 RA Manager Field Mapping

Maps these fields in /suspended_usage/telco	To these fields in /process_audit/batchstat/status
PIN_FLD_BYTES_IN	PIN_FLD_VOLUME_RECEIVED

Table 6-7 (Cont.) RA Manager Field Mapping

Maps these fields in /suspended_usage/telco	To these fields in /process_audit/batchstat/status
PIN_FLD_BYTES_OUT	PIN_FLD_BYTES_VOLUME_SEND
PIN_FLD_BYTES_CALL_DURATION	PIN_FLD_BYTES_EVENT_DURATION
PIN_FLD_BYTES_EDR_COUNT	PIN_FLD_BYTES_EDR_COUNT

PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY sets and returns errors if there are any objects other than **/suspended_usage/telco** objects in the input flist.

You can change the behavior of PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY to read and aggregate any fields of the **/suspended_usage** storable class and its subclasses. The opcode can then map this data to corresponding fields in the **/process_audit/batchstat/status** storable class.

7

Charging Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) charging opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Subscription Rating Opcodes
- Implementing Custom Rating
- Getting ERAs for an Event
- Modifying Rated Events
- Modifying ERAs
- Managing Credit Limits and Sub-Balance Consumption Rules
- Maintaining Subscriber's Charging Preferences Data
- Modifying Events before Sending Them to ECE
- Rerating Opcodes
- Suspense Management
- Provisioning Process Opcode Flow
- GPRS Opcodes

Opcodes Described in This Chapter

Table 7-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 7-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_SPEC_RATES	Implementing Custom Rating
PCM_OP_ACT_USAGE	Resubmitting Suspended Batches Editing Suspended Records in Bulk Writing Off Suspended Records in Bulk Deleting Suspended Records in Bulk

Table 7-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES	Deleting Records for Suspended Batches
PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES	Resubmitting Suspended Batches
PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES	Writing Off Suspended Batches
PCM_OP_BILL_GET_LIMIT	Retrieving Credit Limits
PCM_OP_BILL_POL_POST_SET_LIMIT_AND_CR	Managing Credit Limits and Sub-Balance Consumption Rules
PCM_OP_BILL_POL_PRE_SET_LIMIT_AND_CR	Managing Credit Limits and Sub-Balance Consumption Rules
PCM_OP_BILL_SET_LIMIT_AND_CR	How BRM Handles Consumption Rules and Credit Limits
PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES	Maintaining Subscriber's Charging Preferences Data
PCM_OP_CUST_MODIFY_PROFILE	About Creating Service Orders for Profile Changes
PCM_OP_CUST_SET_PASSWD	Implementing Custom Rating
PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES	Maintaining Subscriber's Charging Preferences Data
PCM_OP_GPRS_APPLY_PARAMETER	About Associating APNs and QoS with GPRS Services Mapping Service Types to Service-Specific Opcodes Associating APN and QoS Pairs with GPRS Services Updating Custom GPRS Service Fields
PCM_OP_GPRS_POL_APPLY_PARAMETER	About Associating APNs and QoS with GPRS Services Associating APN and QoS Pairs with GPRS Services Updating Custom GPRS Service Fields
PCM_OP_GSM_APPLY_PARAMETER	Updating Custom GSM Service Fields
PCM_OP_GSM_POL_APPLY_PARAMETER	Updating Custom GSM Service Fields
PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT	Modifying Events before Sending Them to ECE Modifying BRM Events That Make Up Account Synchronization Business Events
PCM_OP_IFW_SYNC_PUBLISH_EVENT	Processing BRM Events That Make Up Account Synchronization Business Events
PCM_OP_PROV_PUBLISH_SVC_ORDER	Provisioning Process Opcode Flow
PCM_OP_PROV_UPDATE_SVC_ORDER	Provisioning Process Opcode Flow
PCM_OP_RATE_AND_DISCOUNT_EVENT	Modifying Rated Events
PCM_OP_RATE_GET_ERAS	Getting ERAs for an Event Modifying ERAs
PCM_OP_RATE_POL_POST_RATING	Modifying Rated Events
PCM_OP_RATE_POL_PRE_RATING	Modifying Rated Events Modifying ERAs
PCM_OP_RATE_POL_PROCESS_ERAS	Getting ERAs for an Event Modifying ERAs
PCM_OP_RERATE_CREATE_JOB	How BRM Creates Rerate Jobs How BRM Handles Duplicate Rerate Jobs

Table 7-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_RERATE_INSERT_RERATE_REQUEST	How BRM Creates Rerate Jobs How BRM Handles Duplicate Rerate Jobs Customizing Automatic Creation of Rerate Jobs Specifying a Rerate Reason Code Specifying Selection Criteria Specifying Price Overrides Configuring Event Notification for Override Pricing
PCM_OP_SUBCRIPTION_RERATE_REBILL	Customizing Event Searches for Rerating Specifying Price Overrides
PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST	How BRM Creates Rerate Jobs Specifying a Rerate Reason Code Customizing Automatic Creation of Rerate Jobs Configuring Event Notification for Override Pricing
PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE	Customizing Event Searches for Rerating
PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE	Rerating Cycle Forward and Cycle Forward Arrears Events
PCM_OP_SUBSCRIPTION_RATE_CHANGE	Rerating Cycle Forward and Cycle Forward Arrears Events
PCM_OP_SUSPENSE_DEFERRED_DELETE	Deleting Suspended Records in Bulk
PCM_OP_SUSPENSE_DELETE_USAGE	Deleting Suspended Records
PCM_OP_SUSPENSE_EDIT_USAGE	Changing the Contents of Fields in Suspended Event Records Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off Deleting Records in a CDR File
PCM_OP_SUSPENSE_RECYCLE_USAGE	Initiating Suspense Recycling
PCM_OP_SUSPENSE_SEARCH_DELETE	Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off Deleting Records in a CDR File Deleting Suspended Records in Bulk
PCM_OP_SUSPENSE_SEARCH_EDIT	Editing Suspended Records in Bulk
PCM_OP_SUSPENSE_SEARCH_RECYCLE	Recycling Suspended Records
PCM_OP_SUSPENSE_SEARCH_WRITE_OFF	Writing Off Suspended Records in Bulk
PCM_OP_SUSPENSE_UNDO_EDIT_USAGE	Undoing Edits to Suspended Event Records
PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE	Writing Off Suspended Records
PCM_OP_TCF_APPLY_PARAMETER	About Associating APNs and QoS with GPRS Services Mapping Service Types to Service-Specific Opcodes
PCM_OP_TCF_CREATE_SVC_ORDER	About Creating Service Orders for Profile Changes
PCM_OP_TCF_POL_APPLY_PARAMETER	About Associating APNs and QoS with GPRS Services Mapping Service Types to Service-Specific Opcodes Associating APN and QoS Pairs with GPRS Services
PCM_OP_TCF_PROV_CREATE_SVC_ORDER	Provisioning Process Opcode Flow
PCM_OP_TCF_PROV_HANDLE_SVC_ORDER	Provisioning Process Opcode Flow

Table 7-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_TCF_PROV_SERVICE_ORDER_NOTIFY	Provisioning Process Opcode Flow
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_ATTR	Provisioning Process Opcode Flow
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE	Provisioning Process Opcode Flow
PCM_OP_TCF_PROV_SIMULATE_AGENT	Provisioning Process Opcode Flow
PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT	Provisioning Process Opcode Flow
PCM_OP_TCF_SVC_LISTENER	About Associating APNs and QoS with GPRS Services

Subscription Rating Opcodes

The following opcodes are used for subscription rating:

- PCM_OP_RATE_EVENT. This opcode applies pricing and taxes to a subscription event (for example, a recurring charge or a purchase charge).

  When the optional timestamp field PIN_FLD_WHEN_T is present in the input flist, the opcode searches for the charge offer that is valid at the time specified in the field. It uses the charge offer to rate the event.

  PCM_OP_RATE_EVENT also locates any rollover objects for events, and returns details of the rollover object to the calling opcode.

  Uses the index value in the **/product** object's PIN_FLD_TAX_SUPPLIER field to locate the correct tax supplier in the cache.
- PCM_OP_RATE_GET_ERAS. This opcode retrieves the extended rating attribute (ERAs) for an event.

  See "Getting ERAs for an Event" .
- PCM_OP_RATE_GET_PRODLIST. This opcode retrieves the list of charge offers owned by an account and filters them with input criteria.
- PCM_OP_RATE_TAX_CALC. This opcode calculates taxes due at the time of purchase or billing.

  See "How PCM_OP_RATE_TAX_CALC Calculates Taxes".
- PCM_OP_RATE_TAX_EVENT. This opcode directs the calculation of taxes for subscription events.

Implementing Custom Rating

You can use the PCM_OP_ACT_POL_SPEC_RATES policy opcode to create a custom rating application. For example, you can rate administrative events or configure rating based on nonevent attributes such as an account attribute or a profile attribute.

You can configure the PCM_OP_ACT_POL_SPEC_RATES policy opcode to be called by another opcode. The opcode that calls the PCM_OP_ACT_POL_SPEC_RATES policy opcode passes in the inherited information for an event. For example, to rate a password change, the PCM_OP_CUST_SET_PASSWD opcode specifies the inherited password information.

To charge for custom events and attributes:

1. Modify the PCM_OP_ACT_POL_SPEC_RATES policy opcode or write a custom policy opcode to analyze the event data and assign the charge and impact category.

   See the policy source file of the PCM_OP_ACT_POL_SPEC_RATES policy opcode for a sample implementation.

2. Edit the **pin_spec_rates** file in *BRM_home***/sys/data/config** to associate the opcode to the event type that it rates.

   The **pin_spec_rates** file includes examples and instructions.

   > **Note:**
   >
   > When you run the **load_pin_spec_rates** utility, it overwrites the existing setup for administrative events charges. If you are updating a set of administrative events charges, you cannot load new charges only. You load complete sets of charges each time you run **load_pin_spec_rates**.

3. Create a configuration file for the **load_pin_spec_rates** utility.

4. Run the **load_pin_spec_rates** utility to load the contents of the **pin_spec_rates** file into the database.

   ```
   load_pin_spec_rates pin_spec_rates_file
   ```

5. Define impact categories for the event in PDC or Pricing Center.

6. In PDC or Pricing Center, create the charge offers that can be used to rate an event that is rated by your customized opcode.

Getting ERAs for an Event

PCM_OP_RATE_GET_ERAS retrieves the extended rating attribute (ERAs) for an event. This opcode does the following:

- Reads the **/profile/serv_extrating** and **/profile/acct_extrating** objects associated with the service and account in the event.
- Reads **/group/sharing/profile** objects to identify profile sharing groups associated with the services or subscription services in the event. If a profile sharing group is found and is active at the time of the event, the opcode reads the **/profile/serv_extrating** or **/profile/acct_extrating** object associated with the group.
- Checks whether each ERA was valid at the time of the event.
- Returns the name of each valid ERA and the names of ERA labels belonging to each valid ERA to the calling policy opcode, PCM_OP_RATE_POL_PROCESS_ERAS.

Modifying Rated Events

Use PCM_OP_RATE_POL_PRE_RATING to modify subscription events before rating. PCM_OP_RATE_POL_PRE_RATING is called by PCM_OP_RATE_AND_DISCOUNT_EVENT.

Use PCM_OP_RATE_POL_POST_RATING to modify rated **/event** objects (for example, change the G/L ID of an event).

The input flist matches the **/event** object that you are modifying. The output flist contains the event field to be changed in the rated object.

This example shows how to change the G/L ID of an event:

```
STORABLE CLASS /event {
   DESCR = "Objects of the event class are created to record the various "
   "system-initiated and user-initiated events. The events are "
   "either related to a specific service or to an account. The "
   "event includes generic information such as start and end "
   "times as well the various charges that are incurred by the "
   "account due to this event.";
   SEQ_START = "1";
   INT32 PIN_FLD_GL_ID {
      DESCR = "GLID associated with this balance impact. "
      "Don't care if 0.";
      ORDER = 0;
      CREATE = Optional;
      MODIFY = Writable;
   }
}
```

Modifying ERAs

Use PCM_OP_RATE_POL_PROCESS_ERAS to modify ERAs. This opcode calls PCM_OP_RATE_GET_ERAS to find the valid service and account ERAs for an event. This opcode then populates the PIN_FLD_USAGE_TYPE field with the names of valid ERAs and populates the PIN_FLD_PROFILE_LABEL_LIST field with ERA label names.

PCM_OP_RATE_POL_PROCESS_ERAS is called by PCM_OP_RATE_POL_PRE_RATING.

PCM_OP_RATE_POL_PROCESS_ERAS returns the output to PCM_OP_RATE_POL_PRE_RATING.

Managing Credit Limits and Sub-Balance Consumption Rules

Sub-balance consumption rules specify the order in which sub-balances are consumed. For example, when a customer is granted minutes with different validity periods, you can specify which minutes to use first based on the validity start time and end time.

Permanent credit limits specify the maximum amount of a balance element, such as US Dollars, that can accumulate in a balance group. When a credit limit is reached, businesses typically deny customers access to the services associated with the balance group. For example, you might set a customer's credit limit to $100 and then deny service after the customer reaches that limit.

Temporary credit limits specify the maximum amount of a balance element that can accumulate in a balance group during a specified validity period. For example, you might set customer B's credit limit to $200 for December 1, 2031 through January 31, 2032. If the customer also has a permanent credit limit for the same balance element, the temporary credit limit amount is appended to the permanent credit limit amount. For example, if customer B has a permanent credit limit of $100, the customer's credit limit would be $300 for December 1, 2031 through January 31, 2032.

You use the following opcodes to manage credit limits and consumption rules in an account:

- To set sub-balance consumption rules, permanent credit limits, temporary credit limits, or automatic top-up thresholds, use PCM_OP_BILL_SET_LIMIT_AND_CR. See "How BRM Handles Consumption Rules and Credit Limits".
- To retrieve the permanent and temporary credit limits for an account, use PCM_OP_BILL_GET_LIMIT. See "Retrieving Credit Limits".

- To modify the validity dates for an account's temporary credit limit, use PCM_OP_BILL_POL_PRE_SET_LIMIT_AND_CR. By default, this policy opcode does nothing.
- To customize the notification sent after a temporary credit limit is created or expires, use PCM_OP_BILL_POL_POST_SET_LIMIT_AND_CR.
- To customize a customer's permanent credit limit, use PCM_OP_CUST_POL_PREP_LIMIT and PCM_OP_CUST_POL_VALID_LIMIT.

  PCM_OP_CUST_POL_PREP_LIMIT is called by PCM_OP_BILL_SET_LIMIT_AND_CR. PCM_OP_CUST_POL_PREP_LIMIT is an empty hook. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

How BRM Handles Consumption Rules and Credit Limits

Use PCM_OP_BILL_SET_LIMIT_AND_CR to set one or more of the following for an account: a temporary credit limit, a permanent credit limit, a sub-balance consumption rule, or an automatic top-up threshold.

The opcode sets the consumption rule and credit limits in a balance group for both currency and noncurrency balances. This opcode must be called separately for each balance to set. PCM_OP_BILL_SET_LIMIT_AND_CR creates a new **/event/billing/limit** event each time a credit limit is changed and a new **/event/billing/consumption_rule** event each time a consumption rule is changed.

By default, PCM_OP_BILL_SET_LIMIT_AND_CR sets or changes the credit limit, consumption rule, and thresholds in an account's default **/balance_group** object. To set a consumption rule, credit limit, credit floor, or threshold for any of the other billing entities associated with the object, specify them with the optional PIN_FLD_BAL_GRP_OBJ input flist field.

- A consumption rule is passed in the PIN_FLD_RULES array of the PCM_OP_BILL_SET_LIMIT_AND_CR input flist, which contains the balance element ID of the balance to change or create. The consumption rule is set in the PIN_FLD_CONSUMPTION_RULE field of the **/balance_group** object of the various billing entities.
- A permanent credit limit is passed in the PIN_FLD_LIMIT array of the opcode's input flist, which contains the balance element ID of the balance to change or create. The credit limit is set in the PIN_FLD_BALANCES array of the **/balance_group** object of the various billing entities. The POID of this object is passed in the PIN_FLD_BAL_GRP_OBJ field of the input flist.

  If the opcode's PIN_FLD_OVERRIDE_CREDIT_LIMIT input flist field is set to **0**, the account's credit limit is enforced. If the flist field is set to **1**, the account's credit limit is ignored and the account's balance can exceed the limit.
- A temporary credit limit is passed in the PIN_FLD_TEMP_LIMIT array of the opcode's input flist, which contains the resource ID and validity dates of the balance to change or create. If validity dates are not passed in, they are set from the current date to never expires.

  The opcode sets the temporary credit limit in the PIN_FLD_SUB_BAL_IMPACTS array of the **/balance_group** object of the various billing entities. If the **/balance_group** object contains another temporary credit limit with matching validity dates, they are merged into one sub-balance.

  When processing temporary credit limits, PCM_OP_BILL_SET_LIMIT_AND_CR calls the following policy opcodes:

  - PCM_OP_BILL_POL_PRE_SET_LIMIT_AND_CR: Allows you to modify the validity dates for a temporary credit limit.
  - PCM_OP_BILL_POL_POST_SET_LIMIT_AND_CR: Generates an **/event/notification/billing/temp_limit** notification event when a temporary credit limit is added, and generates an **/event/notification/billing/temp_limit/expiry** notification event when a temporary credit limit has expired. You can use this policy opcode to customize the **/event/notification/billing/temp_limit** and **/event/notification/billing/temp_limit/expiry** notification events.

- A credit floor value is passed in the PIN_FLD_CREDIT_FLOOR field in the PIN_FLD_LIMIT array in the opcode's input flist. Alternatively, you can use the PIN_FLD_DYNAMIC_CREDIT_FLOOR field to determine the credit floor using the granted amounts from the sub-balances that are valid for the current cycle for the resource (for example, minutes). If PIN_FLD_DYNAMIC_CREDIT_FLOOR is set to **1**, PIN_FLD_CREDIT_FLOOR is set to NULL, regardless of any value for that parameter that is passed.
- An automatic top-up threshold is passed in the PIN_FLD_RESRC_FIXED_THRESHOLD or PIN_FLD_RESRC_PERC_THRESHOLD field of the PIN_FLD_LIMIT array of the opcode's input flist. The thresholds are stored in the **/config/credit_profile** object tied to the balance group.

For nonpaying child accounts belonging to a PR_RTCE balance monitoring group: If PIN_FLD_ROLL_UP_CREDIT_LIMIT is set to **1**, the opcode rolls up the account's credit limit to the owner of the PR_RTCE balance monitoring group.

A noncurrency balance without an explicitly set credit limit has a default credit limit of zero. If PCM_OP_BILL_SET_LIMIT_AND_CR is called for a nonexistent balance, the balance is created, the credit limit is set, and the current balance is set to zero.

PCM_OP_BILL_SET_LIMIT_AND_CR does not affect the current balances of the account.

To return all fields in the event object, set the PCM_OPFLG_READ_RESULT flag. If this flag is not set, PCM_OP_BILL_SET_LIMIT_AND_CR returns only the POID.

To run PCM_OP_BILL_SET_LIMIT_AND_CR without changing data in the database, use the PCM_OPFLG_CALC_ONLY flag.

- If the flag is set, no fields in the database are changed and the event object is not created. The fields that would have been used to create the event object are returned to the caller.
- If the flag is *not* set, either the **/event/billing/limit** object or the **/event/billing/consumption_rule** object is created to record details of the operation.

Retrieving Credit Limits

Use PCM_OP_BILL_GET_LIMIT to retrieve the permanent and temporary credit limits for an account. To do so, pass in the account POID.

To retrieve credit limits based on specific criteria, also include the following optional input:

- A resource ID, such as **840** for US Dollars, to retrieve credit limits for a specific resource type.
- A balance group POID to retrieve credit limits for a specific balance group.
- A timestamp, such as **1956556800** for Jan 1 00:00:00 2032, where the timestamp falls between a credit limit's validity start date and validity end date (that is, validityStartDate < timestamp < validityEndDate). For example, if the timestamp is **1956556800**, the opcode

could return a credit limit with validity dates from December 15, 2031 through January 15, 2032, or with validity dates from December 29, 2031 through January 29, 2032.

The opcode returns details about the account's permanent credit limits in the PIN_FLD_LIMIT output flist array, and temporary credit limits in the PIN_FLD_TEMP_LIMIT output flist array.

Maintaining Subscriber's Charging Preferences Data

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

- PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES

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

  If a name for the profile object is not provided for PIN_FLD_NAME in the input flist, this opcode creates a profile with the default name PIN_Profile_Object.

  If the PIN_FLD_POID field in the input flist contains the complete POID of the **/profile/subscriber_preferences** object, this opcode modifies the data in the **/profile/subscriber_preferences** object.

  If the PIN_FLD_DELETED_FLAG field in the input flist is set to **1**, PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES deletes the **/profile/subscriber_preferences** object.

- PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES

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

  If the PIN_FLD_SUBSCRIBER_PREFERENCE_ID field is populated in the input flist, this opcode returns the data from the **/profile/subscriber_preferences** object for that preference only. Otherwise, this opcode returns all the preferences for the account.

Modifying Events before Sending Them to ECE

You can modify the BRM events that make up a business event before the business event is sent to Oracle DM for publishing to ECE. You do this by customizing the account synchronization policy opcode, PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT.

PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT modifies specified triggering events before they are included in a business event. This opcode can also be used to filter out certain events not included in account synchronization (for example, an event that has balance impacts only for currency balances), thereby reducing the traffic in the queue.

If you pass an event that does not need modification, PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT passes it directly to the Payload Generator External Module (EM) without modification.

You might want to modify an event to filter out unneeded data, which can improve performance if you publish large quantities of events. You can also use a flag to specify how adjustments should be applied to the account balance (for example, to permit the account to have a negative balance).

To modify a business event before sending it to ECE:

1. Ensure that all the BRM events that make up the business event are associated with opcode number 3626 (PCM_OP_IFW_SYNC_PUBLISH_EVENT) in your system's event notification list.
2. Use PCM_OP_IFW_SYNC_PUBLISH_EVENT to process the BRM events that make up the business event.

   See "Processing BRM Events That Make Up Account Synchronization Business Events".
3. Use PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT to customize the BRM events that make up the business event.

   See "Processing BRM Events That Make Up Account Synchronization Business Events".

Processing BRM Events That Make Up Account Synchronization Business Events

PCM_OP_IFW_SYNC_PUBLISH_EVENT passes BRM events that make up account synchronization business events to PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT for modification.

PCM_OP_IFW_SYNC_PUBLISH_EVENT is called by the event notification feature when an event associated with this opcode in your system's event notification list occurs.

By default, PCM_OP_IFW_SYNC_PUBLISH_EVENT does not modify events; it only passes them to the policy opcode.

If the BRM event is published in the business event, PCM_OP_IFW_SYNC_PUBLISH_EVENT returns one of the following POIDs:

- If the object passed in was an **/event** type object, the event POID is returned.
- If the object passed in was *not* an **/event** type object, a POID of type **/publish** is returned.

If the BRM event is not published (for example, if you filter out the event based on your business logic), the input flist is returned.

Modifying BRM Events That Make Up Account Synchronization Business Events

You can customize PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT to remove or enhance BRM events and event fields before they are assembled into business events and sent to ECE. For example, you can customize this opcode to do the following:

- Filter out BRM events that you do not want to include in published business events for various business or performance reasons.

> **Note:**
>
> – Do not filter out balance impact fields based on the balance element ID. If you do, you might remove fields needed by ECE.
>
> – Do not modify the code that prepares the login fields (the **fm_ifw_sync_pol_prep_logins** function). This code is required when a login is changed.

- Modify the value of the PIN_FLD_FLAGS field that is added to PIN_FLD_BAL_IMPACT arrays. This field specifies how to apply adjustments. For example, you can permit a negative balance by specifying a value of **4**. If you permit a negative balance, the negative amount is deducted from the account balance with the next billing cycle.

  For example, you might use this method when a customer purchases a service that includes 60 minutes per month and cancels the service before the end of the month. If the customer used all 60 minutes but you prorate the minutes, the amount of usage for the canceled period can be deducted with the next billing cycle (provided the customer has a positive balance at that time).

If the BRM event is published in the business event, PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT returns one of the following POIDs:

- If the object passed in was an **/event** type, the event POID is returned.
- If the object passed in was *not* an **/event** type object, a POID of type **/publish** is returned.

If the BRM event is not published (for example, if you filter out the event based on your business logic), the input flist is returned.

Rerating Opcodes

See the following topics:

- How BRM Tracks Rerated Events
- How BRM Creates Rerate Jobs
- How BRM Handles Duplicate Rerate Jobs
- Rerating Cycle Forward and Cycle Forward Arrears Events
- Customizing Event Searches for Rerating
- Customizing Automatic Creation of Rerate Jobs

How BRM Tracks Rerated Events

To back out and rerate events, BRM generates adjustment events (**/event/billing/adjustment/event**). These events contain all of the balance impacts in the original events to fully negate the events. When an event is rerated, the PIN_FLD_RERATE_OBJ field in the original rated event references this backout adjustment event.

When usage events are rerated, the new balance impacts of rerating are included in the same adjustment event that backs out the original balance impacts. This means the original event references the new event that contains the rerated balance impacts.

When cycle fee, cycle discount, fold, and rollover events are rerated, new cycle events are created to apply the balance impacts of rerating. In this case, there is no relationship between the original events and the new (cycle) events that contains the rerated balance impacts.

Because you can use flexible cycles and multiple discounts, there might be more than one rollover, fold, and discount event to rerate for a given billing cycle:

- A fold event is generated per balance for each balance that has a fold.
- A rollover event is generated per balance group for each charge offer that has a rollover.
- A discount event is generated for each cycle discount.

How BRM Creates Rerate Jobs

A rerate job is a pair of **/job_batch/rerate** and **/job/rerate** objects in the BRM database. There is a one-to-one relationship between **/job/rerate** objects and **/job_batch/rerate** objects, and both are created in the same transaction when one or more accounts are selected for rerating.

BRM uses the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode to create rerate jobs. This opcode is called by the following:

- The **pin_rerate** utility, which can be used to manually create rerate jobs
- BRM's automatic rerate job creation mechanism
- Your custom rerating configuration

PCM_OP_RERATE_INSERT_RERATE_REQUEST receives the following information in the input flist:

- The **/account** POIDs to rerate (required).
- The time from which events must be rerated for the accounts (required).
- Accounts related to the accounts to be rerated (for example, an account that used the account's service before a line transfer) along with the start time for each account.

> **Note:**
>
> If more than one account is included in the rerate job, the same rerate start time, selection criteria, and price overrides apply to all accounts.

- Your rerate reason (the default rerate reason passed in for BRM automatic rerating scenarios is **0**).
- Your selection criteria to identify the events to rerate.
- A list of charge offers, discount offers, or bundles to override the subscriber's existing pricing components during rerating.

To create rerate jobs, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following:

1. Determines whether to check for duplicate rerate job requests by using the PIN_RERATE_FLG_MULTI_ACCT flag:
   - If the flag is *set*, the opcode skips the duplicate check.
   - If the flag is *not set*, the opcode performs the duplicate check. See "How BRM Handles Duplicate Rerate Jobs".

   > **Note:**
   >
   > The PIN_RERATE_FLG_MULTI_ACCT flag is set when you run the **pin_rerate** utility with the **-G** *groupOwnerPoid* parameter. See "pin_rerate" in *BRM Rerating Events*.

2. Creates the **/job/rerate** and **/job_batch/rerate** objects.

BRM creates rerate jobs as follows:

1. An event occurs that requires accounts to be rerated.
2. The opcode that triggers rerate job creation when that particular event occurs creates a notification event.
3. The event notification mechanism calls PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST.
4. PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST takes as input the event type and the rerate reason code associated with the event and analyzes the event to determine if rerating is required.

   > **Note:**
   >
   > BRM reserves the range of 100 to 120 for automatic rerating reason codes.

5. If rerating is required, PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST calls PCM_OP_RERATE_INSERT_RERATE_REQUEST along with the appropriate rerating criteria to create a rerate job.

   PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST takes as input the event type and the rerate reason code associated with the event. It analyzes the event to determine if rerating is required. If rerating is required, it sets the rerate reason code to **0** and calls PCM_OP_RERATE_INSERT_RERATE_REQUEST with the appropriate rerating criteria to create a rerate job.

   PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST calls PCM_OP_SUBSCRIPTION_INSERT_RERATE_REQUEST only when the reason code passed in is one of those reserved for automatic rerating (reason codes between 100 and 120). By default, if a different reason code is passed in, the opcode does nothing. To create rerate jobs for reason codes other than those reserved for automatic rerating, you must customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST.

   By default, PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST passes a reason code of **0** (no rerate reason) to PCM_OP_SUBSCRIPTION_INSERT_RERATE_REQUEST. To rerate accounts separately based on a rerate reason code, customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST to pass the rerate reason code it receives (or a different rerate reason code). The rerate reason code is stored in the rerate job object, and you can select rerate jobs based on the specific rerate reason code when you run **pin_rerate** to process rerate jobs.

   PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST only handles rerating events for BRM's default automatic rerating scenarios. If you do not want to rerate events for those scenarios or only want to rerate events for a few of them, you can customize this opcode to achieve those results.

   You can also customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST in other ways. For example, you can change whether rerate jobs are created for specific events: such as when you want to rerate events for only a few (rather than all) automatic rerating scenarios.

   PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST is not called by any opcode.

How BRM Handles Duplicate Rerate Jobs

PCM_OP_RERATE_INSERT_RERATE_REQUEST checks for existing jobs when a new rerate request is made. This opcode compares the data for each top-level account in the rerate request to the data in the existing jobs. To avoid duplicate rerate jobs, accounts are either eliminated from or updated in the rerate request or the existing jobs.

After the duplicate detection process is complete for each top-level account in the rerate request, PCM_OP_RERATE_INSERT_RERATE_REQUEST creates a rerate job with the resulting contents of the rerate request. It then performs checks on the sub-accounts in the rerate request to check for duplicate jobs for line transfers before creating the rerate job.

When the rerate reason code and rerate selection criteria for a rerate request match those of one or more rerate jobs, the request is considered a possible duplicate. When the rerate reason code and rerate selection criteria do not match, the rerate request is not a duplicate, and a new rerate job is created.

To avoid duplicate rerate jobs for an account, PCM_OP_RERATE_INSERT_RERATE_REQUEST compares the following values for duplicate rerate job requests:

- The rerate start times. BRM always uses the earlier start time to ensure that all events are rated.
- The price overrides (if any). BRM uses the latest price overrides for an account because it is assumed to be the most updated.

When the price overrides match, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following for each top-level account:

- If the rerate start time of the rerate request is later than or equal to the start time of the existing job:
  - If there is only this account in the rerate request, deletes the request. No rerate job is needed.
  - If there are many accounts in the new request, removes only that account from the new request.
- If the rerate start time of the rerate request is earlier than the start time of the existing job:
  - If the existing job contains only this account, updates the existing job to use the earlier start time and removes the account from the rerate request.
  - If the existing job contains other accounts, removes the account from the existing job, keeping the account in the rerate request.

When the price overrides do not match, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following for each top-level account:

- If the rerate start time of the rerate request is the same or earlier than the start time of the existing job:
  - If there is only one account in the existing job, deletes the job.
  - If there are many accounts in the existing job, keeps the account in the rerate request so the new rerate job start time and price overrides are used. Removes the account from the existing job.
- If the rerate start time of the rerate request is later than the start time of the existing job:
  - If there is only one account in the existing job, deletes the job.

- If there are many accounts in the existing job, keeps the account in the rerate request so the new price overrides are used. Updates the rerate request start time to use the existing job's start time. Removes the account from the existing job.

Table 7-2 summarizes how PCM_OP_RERATE_INSERT_RERATE_REQUEST handles duplicate rerate job requests:

Table 7-2 Request Handling by PCM_OP_RERATE_INSERT_RERATE_REQUEST

If the Start Time of the New Request Is:	And the Price Overrides:	Perform This Action:
Equal to or later than that of the existing job	Match	If there is only this account in the rerate request, delete the request. No rerate job is needed. If there are many accounts in the rerate request, remove only that account from the rerate request.
Earlier than that of the existing job	Match	If there is only one account in the existing job: • Update the existing job to use the earlier start time. • Remove the account from the rerate request. If there are many accounts in the existing job: • Keep the account in the rerate request. • Remove the account from the existing job.
Later than that of the existing job	Do not match	If there is only one account in the existing job, delete the job. If there are many accounts in the existing job: • Keep the account in the rerate request so the new price overrides are used. • Update the rerate request job start time to use the existing job's start time. • Remove the account from the existing job.
Equal to or earlier than that of the existing job	Do not match	If there is only one account in the existing job, delete the existing job. If there are many accounts in the existing job: • Keep the account in the rerate request so the rerate request start time and price overrides are used. • Remove the account from the existing job.

Rerating Cycle Forward and Cycle Forward Arrears Events

To rerate cycle forward and cycle forward arrears events, BRM uses PCM_OP_SUBSCRIPTION_RATE_CHANGE. This opcode uses the event notification mechanism to trigger the creation of rerating requests when there is a cycle-forward or cycle-forward-arrears event pricing change in the middle of the current cycle.

When you change a cycle fee by adding a new price tier in PDC or Pricing Center, the event notification feature triggers this opcode. This opcode reads the charge offers associated with the event and creates a **/rate_change** object, which is used by the **pin_rate_change** utility to create rerating requests. Rerating requests are used to create rerate jobs that are processed when you run the **pin_rerate** utility.

When you run the **pin_rate_change** utility after a pricing change, the utility calls PCM_OP_SUBSCRIPTION_RATE_CHANGE and provides details about the accounts and charge offers affected by the pricing change.

PCM_OP_SUBSCRIPTION_RATE_CHANGE returns a notification event of type **/event/notification/rate_change** for each account picked up by the **pin_rate_change** utility.

Depending on how automatic rerating is configured, the notification event triggers the creation of rerating requests.

When you rerate cycle forward and cycle forward arrears events, you must track pricing changes. To do so, BRM uses PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE.

This opcode creates the **/rate_change** object, which stores details about the charge offers affected by a pricing change.

When you change a cycle fee by adding a new price tier, BRM event notification triggers this opcode. This opcode reads the charge offers associated with the event and creates a **/rate_change** object, which is used by the **pin_rate_change** utility to create rerating requests.

Customizing Event Searches for Rerating

You can further refine the event selection criteria used for rerating by customizing PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE. This opcode is called when the **pin_rerate** utility is run with **-r** parameter to indicate selective rerating.

> **Note:**
>
> An alternative to customizing this opcode to filter the events selected for rerating is to create custom **pin_rerate** parameters instead.

This opcode is called by PCM_OP_SUBCRIPTION_RERATE_REBILL.

PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE receives an event search template that is based on the account and event selection criteria specified in the **pin_rerate** command line: for example, to select events related to services specified by the **-s** parameter.

By default, PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE does not change the search template and returns a copy of the input flist in the output flist.

You can customize the search template in the input flist to rerate specific types of events. Most customizations include changes only to the fields listed in Table 7-3:

Table 7-3 Fields to Customize

Field	Description
PIN_FLD_TEMPLATE	The modified search template.
PIN_FLD_ARGS	The list of search arguments. **Note:** • This list must match the list of arguments in PIN_FLD_TEMPLATE. • It is preferable to have arguments of one type only. For example, an event search based on charge offer objects.

PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE receives the following fields in the RESULTS array from PCM_OP_SUBSCRIPTION_RERATE_REBILL:

- PIN_FLD_POID
- PIN_FLD_CREATED_T
- PIN_FLD_EFFECTIVE_T

- PIN_FLD_END_T
- PIN_FLD_SERVICE_OBJ
- PIN_FLD_ACCOUNT_OBJ
- PIN_FLD_UNRATED_QUANTITY
- PIN_FLD_RERATE_OBJ
- PIN_FLD_BAL_IMPACTS [*]
- PIN_FLD_SUB_BAL_IMPACTS [*]

> **Note:**
>
> – To assure that the existing mandatory fields in the array are passed back, avoid customizing the RESULTS array. Extra fields added to the array are retrieved but ignored by the standard opcode.
>
> – Test your customized template to ensure that it works properly before using it with a working database. Use the **-e** and **-c** parameters with the **pin_rerate** utility to test your modifications.

PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE returns the search template that PCM_OP_SUBSCRIPTION_RERATE_REBILL uses to find events in selected accounts that need rerating.

PCM_OP_SUBSCRIPTION_RERATE_REBILL rerates the events for accounts identified by the **pin_rerate** utility, rerating one account at a time. The rerating start time is specified on the input flist. This opcode calls other opcodes to perform rerating functions.

Customizing Automatic Creation of Rerate Jobs

You can customize automatic creation of rerate jobs. For example, if you rerate usage following a charge offer cancellation, you can set up a charge offer cancellation event to automatically create a rerate job.

To customize automatic creation of rerate jobs, you set up event notification to call a custom opcode that analyzes events to determine if rerating is required. For example, the criteria for rerating might be:

- If the event is a cancel event.
- If the cancel event for a charge offer requires proration on cancellation.
- If rerating needs to occur to calculate proration for this charge offer.

To customize automatic creation of rerate jobs, you must write custom code to specify when to create rerate jobs automatically when certain events occur. You can do one of the following:

- Create one or more custom opcodes. You can create multiple custom opcodes to handle rerating events for different scenarios or create one opcode to handle all scenarios.
- Modify PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST.

Customized automatic rerating works as follows:

1. An event that you have configured in event notification occurs to call the custom opcode. All the fields in the event are passed as input to the custom opcode.

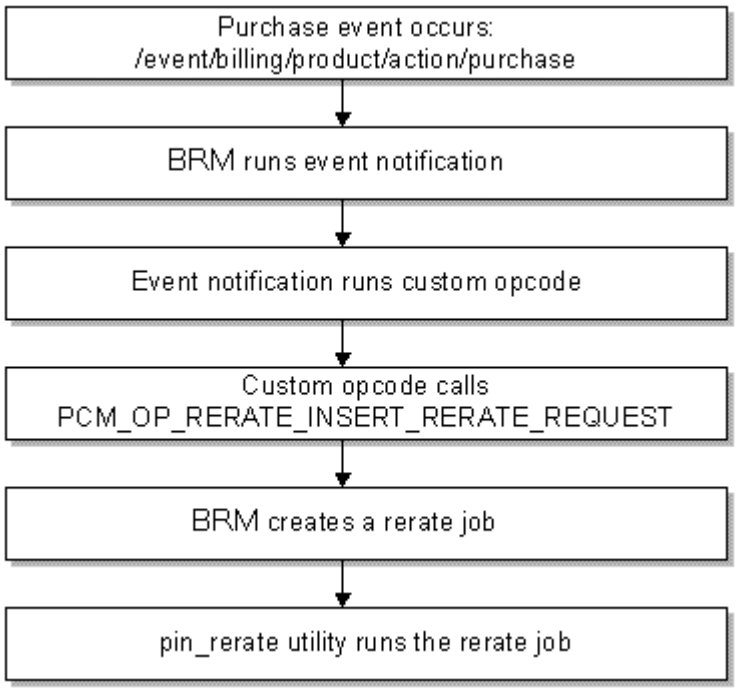
2. The custom opcode analyzes the event using your custom selection criteria to determine if it should trigger rerating.
3. If the event creates a rerate job because it matches your selection criteria, the custom opcode assigns an optional rerate reason code for rerating those events, and specifies any price overrides.
4. PCM_OP_RERATE_INSERT_RERATE_REQUEST creates a rerate job that includes:
   - The account that needs to be rerated.
   - The events that must be rerated for that account.
   - The start time from which to rate the events.
   - The rerate reason.
   - Price overrides, if any.

> **Note:**
>
> If more than one account is included in the rerate job, the same selection criteria, price overrides, and start time apply to all the accounts.

5. You run the **pin_rerate** utility to run the rerate job and rerate the events.

Figure 7-1 shows the rerating process when a purchase event occurs:

Figure 7-1 Customized Automatic Rerating Process



Creating a Custom Opcode for Rerating

Your custom code is required to pass the following to PCM_OP_RERATE_INSERT_RERATE_REQUEST:

- The POIDs of the accounts to be rerated.

  You can also pass in an optional array of additional accounts that are related to these accounts (for example, an account that used the account's service before a line transfer) and the start time for each account.

  > **Note:**
  >
  > If an array of accounts is passed in, the same selection criteria and price overrides apply to all the accounts.

- The time from which events must be rerated for the accounts.

Your custom code can optionally pass in the following to PCM_OP_RERATE_INSERT_RERATE_REQUEST:

- A rerate reason code to take advantage of rerating according to type of rerate job. See "Specifying a Rerate Reason Code".
- Selection criteria so that only those events that are required for rerating are identified for an account. See "Specifying Selection Criteria".
- Price overrides to substitute an account's subscribed pricing for alternate pricing during rerating. See "Specifying Price Overrides".

When an event triggers your custom opcode, your opcode may need to obtain data from other events in the database to obtain all of the information required for rerating.

Specifying a Rerate Reason Code

PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST or your custom opcode can pass an optional rerate reason code to PCM_OP_RERATE_INSERT_RERATE_REQUEST. The rerate reason code is stored in the rerate job and can be used to select jobs from the rerate queue for rerating using **pin_rerate**.

Your rerate reason codes can be any integer, except 1 (1 is reserved for internal use).

> **Note:**
>
> Because you can use multiple rerate reason codes to group rerate jobs for different rerating scenarios, ensure that your rerate reason codes are unique.

The default reason code is **0**.

Specifying Selection Criteria

When an account needs to be rerated, not every event for that account may need to be rerated. Your custom opcode can optionally send selection criteria to PCM_OP_RERATE_INSERT_RERATE_REQUEST to select only the subset of events that needs to be rerated.

For example, you may need to rerate events only when they are generated by a specific service or only when they are rated by a specific charge offer or discount offer. You can specify selection criteria to rerate all accounts' events that have an event type of **/event/delayed/**

session/telco/gsm/telephony but only when they are for the service type **/service/telco/gsm/telephony** and only when they are rated by the charge offer GSM Voice Telephony.

Your selection criteria can be any of the following:

- Event types (array PIN_FLD_EVENTS)
- Service types (array PIN_FLD_SERVICES)
- **/product** object POIDs (array PIN_FLD_PRODUCTS)
- **/discount** object POIDs (array PIN_FLD_DISCOUNTS)
- **/deal** object POIDs (array PIN_FLD_DEALS)

> **Note:**
>
> Charge offers and discount offers are mutually exclusive with bundles. Bundles include charge offers and discount offers, so you can pass in a bundle array instead of the charge offer array and discount offer array.

Specifying Price Overrides

You can specify charge offers, discount offers, or bundles to use for a specific rerate job to override an account's subscribed charge offers, discount offers, or bundles during rerating. The price override applies only to that rerate job; subsequent rerate jobs use the subscribed pricing or their own override pricing.

Your custom opcode can send the following price override data to PCM_OP_RERATE_INSERT_RERATE_REQUEST to set a price override for a rerate job:

- An optional array of override charge offer POIDs along with the charge offer it is overriding (PIN_FLD_OVERRIDE_PRODUCTS).
- An optional array of override discount offer POIDs along with the discount offer it is overriding (PIN_FLD_OVERRIDE_DISCOUNTS).
- An optional array of override **/deal** object POIDs along with the bundle it is overriding (PIN_FLD_OVERRIDE_DEALS).

> **Note:**
>
> Charge offers and discount offers are mutually exclusive with bundles because bundles include charge offers and discount offers. The bundle is translated to a list of charge offers and discount offers when it is passed to the rerating opcodes.

The override charge offers and discount offers or bundles must already be defined in your BRM database. They must also be functionally equivalent to the subscribed charge offers and discount offers or bundles; for example, the priority or service-event map would be the same. In addition, override charge offers and discount offers cannot be configured as first usage charge offers.

> **Note:**
>
> You must rerate accounts when an override pricing charge offer is canceled in a given billing cycle so that refunds can be applied correctly.

BRM creates a record when rerating is run with price overrides by using the **/rerate_session/override_info** object. When override pricing is passed to PCM_OP_SUBSCRIPTION_RERATE_REBILL, this opcodes creates a **/rerate_session/override_info** object to capture the override pricing information used during rerating.

Configuring Event Notification for Customized Automatic Rerating

When you configure event notification for rerating, you specify the following in the **pin_notify** file in *BRM_home***/sys/data/config** (or your own merged event notification configuration file):

- The events that trigger rerating.
- The custom opcodes you want BRM to use to analyze the events, to identify whether rerating is required.

To configure event notification for rerating, do the following:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes the events you want to trigger rerating and the opcode number of the custom opcode you want to analyze those events.

   For example, these entries call the custom opcode with opcode number 10000:

   ```
   # Event notification for rerating
   10000    0    /event/customer/status
   10000    0    /event/billing/product/action/purchase
   ```

3. (Optional) Add, modify, or delete entries in your final event notification list.
4. (Optional) Create custom code for event notification to trigger.
5. Load your final event notification list into the BRM database.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Configuring Event Notification for Override Pricing

If you use override pricing, you must customize automatic rerating to rerate accounts for cases where a refund could not be applied because an override pricing charge offer was canceled in a given billing cycle.

For a given billing cycle, if you rerate an account with an override charge offer and the charge offer is canceled, BRM cannot calculate a refund for that account because the charge offer is not available to apply the necessary cycle fees. When the refund cannot be applied, BRM creates the notification event **/event/notification/product/cancel/no_refund**.

To calculate the correct refund amount, you must customize rerating as follows:

1. Write custom code to specify:
   - The account to rerate from the **/event/notification/product/cancel/no_refund** event.
   - The charge offer to use to apply the cycle fees for the refund to use the override charge offer that was canceled.

BRM uses the base charge offer associated with the account if you do not specify the override charge offer that was canceled. To specify the override charge offer that was canceled, obtain its information from the latest **/rerate_session/override_info** object created for the given account.

2. Include your code in a custom opcode or in PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST that handles automatic rerating (opcode number 3787).

   If you create a custom opcode, it must call PCM_OP_RERATE_INSERT_RERATE_REQUEST to create the rerate job.

3. Set up event notification for the event **/event/notification/product/cancel/no_refund** to call your custom opcode or the policy opcode. For example, to call PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST, you would enter:

```
# Event notification for automatic rerating
3787    0    /event/notification/product/cancel/no_refund
```

Suspense Management

See the following topics:

- Recycling Suspended Records
- Initiating Suspense Recycling
- Changing the Contents of Fields in Suspended Event Records
- Undoing Edits to Suspended Event Records
- Deleting Suspended Records
- Writing Off Suspended Records
- Processing Suspended Records in Bulk
- Suspended Call Record States

Suspended Call Record States

As suspended records are processed by Suspense Manager, they are assigned one of the following states:

- **Suspended**. The record could not be processed by Offline Mediation Controller.
- **Recycling**. The record is being sent through Offline Mediation Controller again to be rated.
- **Succeeded**. The record has been successfully recycled and rated.
- **Written off**. The record will not be recycled, but is being stored for further use.

Table 7-4 lists the details about the states:

Table 7-4 Suspended Call Record States

State	PIN_FLD_STATUS value in /suspended_usage	Can be edited	Can be recycled	Can be written off	Can be deleted	Can be archived and deleted
Suspended	**0**	**Yes**	**Yes**	**Yes**	No	No
Recycling	**1**	No	No	No	No	No

Table 7-4 (Cont.) Suspended Call Record States

State	PIN_FLD_STATUS value in /suspended_usage	Can be edited	Can be recycled	Can be written off	Can be deleted	Can be archived and deleted
Succeeded (Successfully processed)	**2**	No	No	No	**Yes**	**Yes**
Written off	**3**	No	No	No	**Yes**	**Yes**

Recycling Suspended Records

PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for and queues suspended call records for recycling. There are many search criteria that you may use to search the records, such as by CDR file or by recycle key.

You can specify the following criteria:

- A recycle key
- A CDR file
- A search template

PCM_OP_SUSPENSE_SEARCH_RECYCLE is usually called by the **pin_recycle** utility, which searches for and deletes call records with a specific recycle key and a status of **Succeeded** (or **Written-off**).

Searching for Records in a CDR File

The BRM standard recycling feature collects and manipulates all calls contained in a CDR file at the same time. PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for a specified CDR file and queues its suspended event records for recycling. If successful, the event records are assigned a status of **Succeeded**.

Searching for Records with a Recycle Key

PCM_OP_SUSPENSE_SEARCH_RECYCLE is used by features that must temporarily delay rating of event records. This opcode searches for all **Suspended** call records containing the recycle key passed in on the input flist. It then queues those call records to be rated at the next opportunity. If successful, the event records are assigned a status of **Succeeded**.

Initiating Suspense Recycling

PCM_OP_SUSPENSE_RECYCLE_USAGE initiates event record recycling. The Suspense Management Center calls this opcode when the user chooses to recycle suspended event records.

PCM_OP_SUSPENSE_RECYCLE_USAGE can operate in test mode. In test mode, the event record is processed normally. The results, however, are not output if the event record is processed successfully.

PCM_OP_SUSPENSE_RECYCLE_USAGE takes as input an array of **/suspended_usage** object POIDs, a list of suspense override reasons, and a value that indicates whether the event records should be recycled in test mode. This opcode then creates an **/admin_action/suspended_usage/recycle** object with that information.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_RECYCLE_USAGE performs these operations:

- Confirms that the suspended event record has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
- Changes the event record status to **Recycling**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is set to **1**.
- Creates an **/admin_action/suspended_usage/recycle** object containing the recycle mode and override reasons (passed in the input flist).
- Adds the POID of the newly created **/admin_action/suspended_usage/recycle** object to the actions array of each **/suspended_usage** object.
- Sets the PIN_FLD_RECYCLE_OBJ field of each **/suspended_usage** object to the **/admin_action/suspended_usage/recycle** object POID.

After event records are recycled, their **/suspended_usage** objects are updated by using the Suspended Event (SE) Loader:

- If an event record is successfully recycled, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If an event record is not successfully recycled, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. In addition, the following fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_SUSPENSE_REASON.
  - The suspense subreason code, PIN_FLD_SUSPENSE_SUBREASON.
- If PCM_OP_SUSPENSE_RECYCLE_USAGE was run in test mode, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. The corresponding test values (PIN_FLD_TEST_SUSPENSE_REASON, PIN_FLD_TEST_SUSPENSE_SUBREASON, and PIN_FLD_TEST_SUSPENSE_ERROR_CODE) are updated to include information about the results of the test recycling.
- The number of times the record has been recycled (in PIN_FLD_NUM_RECYCLES) is increased by 1.

PCM_OP_SUSPENSE_RECYCLE_USAGE returns the routing POID specified in the input flist.

Resubmitting Suspended Batches

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES resubmits suspended CDR file. Suspense Management Center calls this opcode when the user chooses to resubmit suspended batches.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES takes an array of **/suspended_batch** object POIDs and a list of suspense override reasons as input. This opcode then creates an **/admin_action/suspended_batch/resubmit** object with that information.

For the whole set of **/suspended_batch** objects specified in the input flist, PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Creates an ADMIN_ACTION object, **/admin_action/suspended_batch/resubmit**, with the override reason, for each **/suspended_batch** object.

- Validates the status of each **/suspended_batch** object (Batch Suspense Record) and updates the status with the result of the resubmission.
- Creates an event Flist (with **/event/notification/suspense/batch_resubmit**) and calls PCM_OP_ACT_USAGE in CALC_ONLY mode.
- Closes the transaction

After CDR file are resubmitted, their **/suspended_batch** objects (Batch Suspense Records) are updated by using the SE Loader:

- The PIN_FLD_NUM_RESUBMISSIONS field in each **/suspended_batch** object is incriminated.
- If a batch is successfully resubmitted, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If a batch is not recycled successfully, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. This will cause all the batches in the resubmission task to be rolled back as well. In addition, these fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_BATCH_SUSPENSE_REASON.
  - The suspense sub-reason code, PIN_FLD_BATCH_SUSPENSE_SUBREASON.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES returns the routing POID specified in the input flist.

Changing the Contents of Fields in Suspended Event Records

Use PCM_OP_SUSPENSE_EDIT_USAGE to change the contents of fields in suspended event records. Suspense Management Center calls this opcode to edit a suspended call record.

> **Note:**
>
> You cannot edit the CDR or event record fields of records in a CDR file that has been suspended by batch suspense and has a Batch Suspense Record.

PCM_OP_SUSPENSE_EDIT_USAGE performs these operations:

- Takes as input an array of **/suspended_usage** object POIDs and an array of the event record fields to be edited, including the old and new values.
- For each event record field to be edited, this opcode:
  - Creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
  - Adds the **/admin_action/suspended_usage/edit** object POID to the top of the **/suspended_usage_edits** object stack. If the stack is full, it removes the oldest POID.
- For each **/suspended_usage** object, this opcode:
  - Confirms that the suspended event record has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
  - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.

  - Updates the modified flag (PIN_FLD_EDITED), which indicates that the field has been edited.

PCM_OP_SUSPENSE_EDIT_USAGE returns an array of the POID of the **/admin_action/suspended_usage/edit** objects it creates.

Undoing Edits to Suspended Event Records

Use PCM_OP_SUSPENSE_UNDO_EDIT_USAGE to undo edits to suspended event records.

This opcode is called by Suspense Management Center to the undo edits. It replaces the value of a field in a suspended call record with the value in that field before the last edit was made.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE performs these operations:

- Searches for the **/suspended_usage_edits** object If a **/suspended_usage_edits** object exists, this opcode reads and locks it.
- Confirms that the POID of the **/admin_action/suspended_usage/edit** object on the input flist matches that of the top POID of the **/suspended_usage_edits** object stack. If the two don't match, returns failure status (PIN_FLD_RESULT is set to **1**) and returns the POID at the top of the **/suspended_usage_edits** object stack.
- Confirms that each suspended call record affected by the edit has a status of **Suspended**.
- Undoes the edit by replacing the existing field values with the values before the last edit was saved (the values referenced by the POID at the top of the **/suspended_usage_edits** object stack).
- Adds the session, service, and date and time details of the undo edit operation in the PIN_FLD_UNDO_DATA array of the **/admin_action/suspended_usage/edit** object.
- Removes the POID of the **/admin_action/suspended_usage/edit** object from the top of the **/suspended_usage_edits** object stack.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE returns a **PIN_FLD_RESULT** value of:

- **0** if the undo edit was successful. Also:
  - The PIN_FLD_POID field contains the POID of the **/admin_action/suspended_usage/edit** object.
  - The PIN_FLD_COUNT field contains the number of records that were processed.
- **1** if the **/admin_action/suspended_usage/edit** POID in the input flist does not match the POID at the top of the **/suspended_usage_edits** object stack.

Deleting Suspended Records

Use PCM_OP_SUSPENSE_DELETE_USAGE to delete suspended event records.

The Suspense Management Center calls this opcode to delete event records. Event records can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_SUSPENSE_DELETE_USAGE takes as input an array of **/suspended_usage** object POIDs to be deleted.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_DELETE_USAGE performs these operations:

- Confirms that the suspended event record has a status of **Succeeded** or **Written off.** The value of the PIN_FLD_STATUS field in the **/suspended_usage** object must be **2** or **3**.

- Deletes the **/suspended_usage** object.
- Generates a **/event/notification/suspense/delete** object that records the deletion.

PCM_OP_SUSPENSE_DELETE_USAGE returns the routing POID specified in the input flist.

Deleting Records for Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES to delete **/suspended_batch** objects (Batch Suspense Records).

> **Note:**
>
> This opcode deletes records, not the files associated with them.

Suspense Management Center calls this opcode to delete Batch Suspense Records (**/suspended_batch** objects). Batch Suspense Records can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES takes as input an array of **/suspended_batch** object POIDs to be deleted.

For each **/suspended_batch** object specified in the input flist, PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Confirms that the suspended batch file has a status of **Succeeded** or **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_batch** object must be **2** or **3**. Otherwise, an error code is generated and the transaction ends.
- Deletes the **/suspended_batch** object.
- Generates an **/event/notification/suspense/batch_delete** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES returns the routing POID specified in the input flist.

Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off

Use PCM_OP_SUSPENSE_SEARCH_DELETE to delete call records with a specific recycle key and a status of **Succeeded** or **Written-off**.

You can specify the following criteria:

- A recycle key
- A CDR file
- A search template

This opcode can also delete a **suspended** call record if PIN_FLD_MODE is set correctly.

To search for and recycle suspended call records containing a specific recycle key, use PCM_OP_SUSPENSE_EDIT_USAGE.

Set the **PIN_FLD_FLAGS** field to either of these values:

- **0**: Directs PCM_OP_SUSPENSE_EDIT_USAGE to delete event records with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
- **1**: Directs PCM_OP_SUSPENSE_EDIT_USAGE to delete event records with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
  - **Suspended**. PCM_OP_SUSPENSE_EDIT_USAGE first writes off, then deletes, these event records

Deleting Records in a CDR File

PCM_OP_SUSPENSE_SEARCH_DELETE is used with the BRM standard recycling feature, which acts on all the calls contained in a single CDR file simultaneously. Using **pin_recycle** with the **-d** parameter deletes all calls in a CDR file that have a status of **Succeeded**. Using **pin_recycle** with the **-D** parameter deletes all calls in a CDR file with a status of **Succeeded** or **Written off**.

PCM_OP_SUSPENSE_SEARCH_DELETE is also used by features that must temporarily delay rating of event records. The error prevents event records from being rated by assigning them a status of **Suspended**. Those features then use PCM_OP_SUSPENSE_EDIT_USAGE to find and recycle the call records. If successful, the event records are assigned a status of **Succeeded**. PCM_OP_SUSPENSE_SEARCH_DELETE then deletes those successfully recycled call records.

Writing Off Suspended Records

Use PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE to write off suspended event records.

When a suspended event record is written off, it can no longer be edited or recycled.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE takes as input an array of **/suspended_usage** object POIDs.

- PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE creates an **/admin_action/suspended_usage/writeoff** object that records the write-off.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE performs these operations:

- Confirms that the suspended event record has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
- Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.
- Generates an **/event/notification/suspense/batch_writeoff** event.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE returns the POID of the **/admin_action/suspended_usage/writeoff** object created.

Writing Off Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES to write off suspended CDR files.

When you write off a suspended CDR file, you can no longer resubmit it, but you can delete it.

The opcode creates an **/admin_action/suspended_batch/writeoff** object that records the write-off and sets the status of the Batch Suspense Record (**/suspended_batch**) to **Written-off**.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES performs these operations:

- Create a transaction if it is not already opened.
- Confirms that the suspended event record has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES takes as input an array of **/suspended_batch** object POIDs.
- Adds the POID of the newly created **/admin_action/suspended_batch/writeoff** object to the array of actions in the **/suspended_batch** object.
- Changes the status of the Batch Suspense Record (**/suspended_batch**) to **Written off**; The value of the PIN_FLD_STATUS field in **/suspended_batch** is changed to **3**.
- Generates an **/event/notification/suspense/batch_writeoff** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES returns the POID of the **/admin_action/suspended_batch/writeoff** object created.

Processing Suspended Records in Bulk

You can use the Suspense Manager opcodes to edit, delete, recycle, and write off a large number of suspended records. For more information, see the following topics:

- Processing Suspended Records in Multiple Steps
- Editing Suspended Records in Bulk
- Writing Off Suspended Batches
- Deleting Suspended Records in Bulk

Processing Suspended Records in Multiple Steps

You can process suspended records in multiple steps by calling the opcodes multiple times to avoid a large database transaction:

1. Specify the number of records to process in each opcode call in a configuration file, and load the file into the **/config/pin_suspense_system_params** storable class.
2. Call the opcode in calc-only mode to retrieve the count and POID range of the records that match your search criteria.
3. Use a simple logic to determine the number of times to call the opcodes depending on the number of records you want each opcode call to process.
4. Call the opcode several times with a set of records each time and consolidate the results returned by each opcode call.

> **Note:**
>
> For each opcode call, you must provide the POID range and the corresponding arguments in the input flist.

Because each operation is performed in multiple steps, if the operation is successful in any of the steps, you get the number of records processed. You also get an error message for the unsuccessful records, which you can display to the user. If any one of the steps fails, the entire step and the following steps are canceled and an error message is returned.

> **Note:**
>
> Suspense Management Center and the **pin_recycle** utility perform suspense management operations in multiple steps by calling the opcodes multiple times.

Editing Suspended Records in Bulk

Use PCM_OP_SUSPENSE_SEARCH_EDIT to perform the same set of edits on a large number of suspended records that meet the criteria you specify.

This opcode makes changes to the records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS, to update the objects in the database.

  The opcode finds the accounts that meet the criteria specified in PIN_FLD_ARGS and updates them with the information in PIN_FLD_VALUES.

  Specify the fields and values to set, along with the POID type of the object, in the input flist. You must update at least one field.

  Use the PCM_OPFLG_ADD_ENTRY flag to create array elements. If the specified array element already exists, this flag is ignored. PCM_OPFLG_ADD_ENTRY cannot be used to create ordinary fields.

  The opcode returns the POID type and count of the object whose fields were updated.
- PCM_OP_ACT_USAGE, to generate the edit event notification.

> **Note:**
>
> You cannot undo edits performed on a large number of records or any edits made before the bulk edit operation.

PCM_OP_SUSPENSE_SEARCH_EDIT follows these steps to edit suspended records:

1. Takes as input the following information:
   - The POID type of the suspended usage class.
   - The search criteria template.
   - The fields and values that must be edited in the object.
   - An array of the event record fields to be edited, including the old and new values.

2. Does one of the following:
   a. If the PCM_OPFLG_CALC_ONLY flag is set, returns the count and the POID range of records that meet the search criteria.
   b. If the PCM_OPFLG_CALC_ONLY flag is not set, searches for the **/suspended_usage_edits** objects.
3. Does one of the following:
   a. If it finds **/suspended_usage_edits** objects, clears all the POIDs of the edit actions stored in the objects.

      > **Note:**
      >
      > After the objects are changed, the current changes or previous changes cannot be undone.

   b. If it does not find a **/suspended_usage_edits** object, creates a new **/suspended_usage_edits** object.
4. For each event record field to be edited, creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
5. For each **/suspended_usage** object, performs the following operations:
   - Verifies that the suspended event record has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.
   - Updates the PIN_FLD_EDITED field to indicate that the field has been edited.

If successful, PCM_OP_SUSPENSE_SEARCH_EDIT generates an edit notification event that includes the administrative action POID of the edit action and returns success along with the count of the objects edited. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Writing Off Suspended Records in Bulk

Use PCM_OP_SUSPENSE_SEARCH_WRITE_OFF to write off all suspended records that meet the criteria you define.

> **Note:**
>
> You cannot edit or recycle suspended records that are written off.

This opcode writes off a large set of suspended records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS to mark a large number of objects in the database as written off.
- PCM_OP_ACT_USAGE to generate the write-off event notification.

PCM_OP_SUSPENSE_SEARCH_WRITE_OFF follows these steps to write off suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be written off.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.
3. If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_writeoff** object that records the write off and returns that object with the count of records written off.
4. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:
   - Verifies that the suspended event record has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
   - Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.

If successful, PCM_OP_SUSPENSE_SEARCH_WRITE_OFF generates a write-off notification event that includes the administrative action POID of the write-off action and returns success along with the number of records written off. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Deleting Suspended Records in Bulk

Use PCM_OP_SUSPENSE_SEARCH_DELETE to delete all suspended records that meet the criteria you define. This opcode is scheduled to run at a later time to ensure revenue assurance.

> **Note:**
>
> You can only delete records that are succeeded or written off.

PCM_OP_SUSPENSE_SEARCH_DELETE deletes a large set of suspended records in one database operation instead of accessing the database for each record. It calls PCM_OP_ACT_USAGE to generate the delete event notification.

PCM_OP_SUSPENSE_SEARCH_DELETE follows these steps to delete suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be deleted.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.

   If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_delete** object that records the deletion.
3. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:

- Verifies that the suspended event record has a status of **Succeeded** or **Written off**. The PIN_FLD_STATUS value in the object must be **2** or **3**.
- Event records with a status of **Succeeded** and those with a status of **Written off** that do not have a deferred duration are deleted immediately.
- If the status of the suspended record is **Written off**, the opcode checks if there is a deferred duration and if the deferred duration is greater than 0. The deferred duration is a parameter defined in the **/config/suspense_params** object.

  > **Note:**
  >
  > The load utility is provided to load the deferred duration parameter.

  If there is a deferred duration, then PCM_OP_SUSPENSE_SEARCH_DELETE will not remove the suspended record from the database but rather change the status of the suspended records from written off to delete pending.
  PCM_OP_SUSPENSE_SEARCH_DELETE will create a schedule object to run PCM_OP_SUSPENSE_DEFERRED_DELETE.
- The schedule object will call PCM_OP_SUSPENSE_DEFERRED_DELETE to be run at the deferred duration time.
- PCM_OP_SUSPENSE_SEARCH_DELETE or PCM_OP_SUSPENSE_DEFERRED_DELETE deletes the object from the suspense db.

4. Generates a **/event/notification/suspense/delete** object that records the deletion for each suspended record that was deleted.

If successful, PCM_OP_SUSPENSE_SEARCH_DELETE generates a delete notification event that includes the administrative action POID of the delete action and returns success along with the number of records deleted. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Provisioning Process Opcode Flow

Services Framework provisioning generates service orders as follows:

1. A customer account creates or modifies a service, device, or profile. This generates a notification event.
2. The event notification system calls the opcode specified in the **/config/notify** object. By default, PCM_OP_TCF_PROV_CREATE_SVC_ORDER is called.
3. PCM_OP_TCF_PROV_CREATE_SVC_ORDER performs the following:
   a. Determines whether the service type passed in the input flist is supported by Services Framework. All telco service types and all service types listed in the **/config/service_framework/permitted_service_types** object are supported.
   b. Determines the *provisioning configuration object* to use.
   c. Determines the *service order configuration object* to use.
   d. Generates the service order business event (**/event/provisioning/service_order/telco/***).

> **Note:**
>
> The service order business event contains “telco" in its name for both telco and nontelco service types, because the common substruct for holding the service order data is at the **/event/provisioning/service_order/telco** level.

4. The event notification system calls the opcode specified in the **/config/notify** object. By default, PCM_OP_TCF_PROV_HANDLE_SVC_ORDER is called.
5. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER performs the following:
   a. Determines whether the service order status is NEW. If it is, the opcode calls PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE to update the service order status to READY.

      PCM_OP_TCF_PROV_SERVICE_ORDER_NOTIFY performs service order state changes. PCM_OP_TCF_PROV_SERVICE_ORDER_SET_ATTR updates a service order object.

   b. Determines whether to call the Network Simulator by reading the **simulate_agent** entry in the CM **pin.conf** file. If **simulate_agent** is set to **1**, the opcode calls PCM_OP_TCF_PROV_SIMULATE_AGENT to simulate the provisioning flow with the CM.

   c. Determines the provisioning mode for the service type by reading the **/config/service_framework/permitted_service_types** object. The default is **Queued**.

   d. Calls PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER with the service order event details and provisioning mode.
6. PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER performs any custom actions before it is passed to **dm_prov_telco**, and then returns to the calling opcode. This opcode dis an empty hook, but you can customize it to override the provisioning mode and modify service order event details.
7. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER calls PCM_OP_PROV_PUBLISH_SVC_ORDER to publish the service order.
8. PCM_OP_PROV_PUBLISH_SVC_ORDER publishes the service order to **dm_prov_telco**.
9. **dm_prov_telco** determines the provisioning mode by reading the PIN_FLD_MODE flist entry. **dm_prov_telco** operates in Queued mode when the entry is **0** and Confirmed mode when the entry is **1**.
   - In Queued mode, **dm_prov_telco** queues the request in-memory and calls PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE to update the service order status to PROVISIONING.
   - In Confirmed mode, **dm_prov_telco** sends the service order to the network provisioning agent through a TCP/IP connection and waits for a response.

After the network provisioning agent returns a response in flist format, Services Framework provisioning performs the following:

1. **dm_prov_telco** sends the response to PCM_OP_PROV_PUBLISH_SVC_ORDER.
2. PCM_OP_PROV_PUBLISH_SVC_ORDER passes the response to PCM_OP_TCF_PROV_HANDLE_SVC_ORDER.
3. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER calls PCM_OP_PROV_UPDATE_SVC_ORDER to update the service order status.

4. PCM_OP_PROV_UPDATE_SVC_ORDER changes the service order status to **Processed** and generates the **/event/provisioning/service_order/telco/*** business event.
5. PCM_OP_TCF_PROV_UPDATE_SVC_ORDER updates the **/event/provisioning/service_order/telco** object with the provisioning response specified in the opcode input flist.
6. The event notification system calls the opcode specified in the **/config/notify** object. By default, PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT is called.
7. PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT updates the service's status and supplementary features.

About Creating Service Orders for Profile Changes

Profiles are added and removed from BRM service objects through the purchase and cancellation of charge offers and bundles. Services Framework provisioning includes profiles in service orders as directed by the provisioning configuration object **/config/provisioning/telco**. PCM_OP_TCF_CREATE_SVC_ORDER determines whether to create a service order based on data that has been added, changed, or removed from this object.

This opcode subscribes to the **/event/notification/profile/pre-modify** and **/event/notification/profile/modify** events generated by PCM_OP_CUST_MODIFY_PROFILE to create service orders for profile changes. Based on the value in the status field of the profile object, a service order is created and stored in the **/event/provisioning/service_order/telco/***service_name* event by capturing changes made to the profile object.

GPRS Opcodes

Use GPRS opcodes to configure and provision GPRS services.

About Associating APNs and QoS with GPRS Services

In BRM, an APN is stored as a device (**/device/apn**). Because the APN is a network device and because a single APN can be used by multiple customers, you cannot map APN devices to GPRS services. Instead, the GPRS service references an APN and its appropriate quality of service (QoS) value based on the charge offer purchased by the customer.

BRM associates APN and QoS pairs with a GPRS service as follows:

1. A customer purchases, cancels, or modifies a bundle or charge offer that includes a GPRS service and provisioning tag.
2. The BRM system generates one of the following business events:
   - **/event/billing/deal/purchase**
   - **/event/billing/production/action/modify**
   - **/event/billing/deal/cancel**
   - **/event/billing/product/action/cancel**
   - **/event/billing/product/action/purchase**
3. The BRM event notification system detects the event and calls the opcode specified in the **/config/notify** object. The default configuration specifies to call PCM_OP_TCF_SVC_LISTENER.
4. PCM_OP_TCF_SVC_LISTENER checks the event's start and end date to determine whether the action is deferred for a future date.

- If the event *is not* deferred, the opcode calls PCM_OP_TCF_APPLY_PARAMETER to update the GPRS service and ERA objects impacted by the charge offer provisioning update.
- If the event *is* deferred, the opcode creates a **/schedule** object for executing PCM_OP_TCF_APPLY_PARAMETER at the scheduled time.

5. PCM_OP_TCF_APPLY_PARAMETER retrieves the service extension and service ERA information from the **/config/telco/gprs** object and passes the information to PCM_OP_TCF_POL_APPLY_PARAMETER.
6. When the service type is **/service/telco/gprs**, PCM_OP_TCF_POL_APPLY_PARAMETER calls PCM_OP_GPRS_APPLY_PARAMETER.
7. PCM_OP_GPRS_APPLY_PARAMETER writes the APN and QoS pairs to the flist's PIN_FLD_APN_ARRAY array.
8. PCM_OP_GPRS_APPLY_PARAMETER calls PCM_OP_GPRS_POL_APPLY_PARAMETER to update information about any custom **/service/telco/gprs** fields.
9. PCM_OP_TCF_APPLY_PARAMETER updates the service extensions and ERAs in the **/service/telco/gprs** object.

To customize how BRM associates APNs and QoS values with GPRS services, see "Associating APNs and QoS with GPRS Services".

Associating APNs and QoS with GPRS Services

See the following topics:

- Mapping Service Types to Service-Specific Opcodes
- Associating APN and QoS Pairs with GPRS Services
- Updating Custom GPRS Service Fields

Mapping Service Types to Service-Specific Opcodes

Use PCM_OP_TCF_POL_APPLY_PARAMETER to update information in the flist's PIN_FLD_SERVICES and PIN_FLD_PRODUCTS array and then pass the information to the appropriate service-specific opcode.

PCM_OP_TCF_POL_APPLY_PARAMETER takes as input the configuration object flist, the service flist, and the inherited information flist and then updates the service flist.

PCM_OP_TCF_POL_APPLY_PARAMETER is called by PCM_OP_TCF_APPLY_PARAMETER.

By default, PCM_OP_TCF_POL_APPLY_PARAMETER returns an empty inherited info flist.

If you added a substruct to a customized service, you can use PCM_OP_TCF_POL_APPLY_PARAMETER to fill in the substruct fields. These fields are updated in the database.

For example, a GSM service (**/service/telco/gsm**) could include a field for the bearer in the configuration object (**/config/telco**) in the service extensions array PIN_FLD_SERVICE_EXTENSIONS. You could use PCM_OP_TCF_POL_APPLY_PARAMETER to add the bearer information through the service extension to update the service flist.

Table 7-5 shows the opcode called for each supported service type:

Table 7-5 Supported Service Type Opcodes

Service Type	Opcode Called
/service/telco/gprs	PCM_OP_GPRS_APPLY_PARAMETER
/service/telco/gsm	PCM_OP_GSM_APPLY_PARAMETER

Associating APN and QoS Pairs with GPRS Services

Use PCM_OP_GPRS_APPLY_PARAMETER to associate APN and QoS pairs with a **/service/telco/gprs** service. This opcode is called by PCM_OP_TCF_POL_APPLY_PARAMETER when processing **/service/telco/gprs** services.

PCM_OP_GPRS_APPLY_PARAMETER reads the bearer service, APN name, and QoS information from the input flist's PIN_FLD_SERVICE_EXTENSIONS array and performs the following:

- If the Bearer service is passed in the input flist, the opcode adds the value to the output flist's PIN_FLD_BEARER_SERVICE field of the PIN_FLD_GPRS_INFO substruct.
- If the APN name and QoS are passed in the input flist, the opcode adds the values to the output flist's PIN_FLD_APN array in the PIN_FLD_INHERITED_INFO substruct.

The opcode then calls PCM_OP_GPRS_POL_APPLY_PARAMETER to perform any customizations. See "Updating Custom GPRS Service Fields".

Updating Custom GPRS Service Fields

Use PCM_OP_GPRS_POL_APPLY_PARAMETER to update custom fields in the **/service/telco/gprs** object. This opcode takes as input the configuration object flist, the service flist, and the inherited information flist from the calling PCM_OP_GPRS_APPLY_PARAMETER.

By default, PCM_OP_GPRS_POL_APPLY_PARAMETER returns the information passed in the input flist. This opcode can be customized to update the service flist by adding values to customized fields.

PCM_OP_GPRS_POL_APPLY_PARAMETER is called by PCM_OP_GPRS_APPLY_PARAMETER.

Updating Custom GSM Service Fields

Use PCM_OP_GSM_POL_APPLY_PARAMETER to update custom fields in the **/service/telco/gsm** object. PCM_OP_GSM_POL_APPLY_PARAMETER updates the service flist by adding values to customized fields for a service. This opcode takes as input the configuration object flist, the service flist, and the inherited information flist from the calling PCM_OP_GSM_APPLY_PARAMETER.

By default, PCM_OP_GSM_POL_APPLY_PARAMETER returns an empty inherited info flist.

If you added a substruct to a customized service, you can use PCM_OP_GSM_POL_APPLY_PARAMETER to fill in the substruct fields. These fields are updated in the database.

For example, a GSM service (**/service/telco/gsm**) could include a field for the bearer in the configuration object (**/config/telco**) in the service extensions array PIN_FLD_SERVICE_EXTENSIONS. You could use

PCM_OP_GSM_POL_APPLY_PARAMETER to add the bearer information through the service extension to update the service flist.

PCM_OP_GSM_POL_APPLY_PARAMETER is called by PCM_OP_GSM_APPLY_PARAMETER. PCM_OP_GSM_APPLY_PARAMETER reads the service extensions from the input flist and adds corresponding GSM service values. The opcode reads the bearer service, APN name, and QoS information from the input flist's PIN_FLD_SERVICE_EXTENSIONS array and performs the following:

- If the Bearer service is passed in the input flist, the opcode adds the value to the output flist's PIN_FLD_BEARER_SERVICE field of the PIN_FLD_GSM_INFO substruct.
- If the APN name and QoS are passed in the input flist, the opcode adds the values to the output flist's PIN_FLD_APN array in the PIN_FLD_INHERITED_INFO substruct.

8

Collections Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) collections opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Customizing Collections Manager
- Managing Overdue Balance Collection
- Managing Promise-to-Pay Agreements
- Managing Collections Groups
- Performing Manual Collections Actions
- Performing System Collections Actions
- Retrieving Collections Information

For more information about collections, see "Understanding Collections Manager" in *BRM Collections Manager*.

Opcodes Described in This Chapter

Table 8-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 8-1 Collections Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_COLLECTIONS_ADD_ACTION	Adding Actions to a Collections Scenario
PCM_OP_COLLECTIONS_ASSIGN_AGENT	Assigning Bill Units to a Collections Agent
PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS	Retrieving Aging Buckets Information
PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION	Deleting an Existing Collections Action
PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE	Deleting an Existing Collections Profile
PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO	Deleting an Existing Collections Scenario
PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS	Getting All Currently Defined Collections Actions
PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES	Getting All Currently Defined Collections Profiles

Table 8-1 (Cont.) Collections Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL	Getting Details of a Collections Scenario
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS	Getting All Currently Defined Collections Scenarios
PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES	Getting a List of Message Templates
PCM_OP_COLLECTIONS_CONFIG_SET_ACTION	Creating or Updating Collections Actions Applying Late Fees
PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE	Creating or Updating Collections Profiles
PCM_OP_COLLECTIONS_EXEMPT_BILLINFO	Exempting Bill Units from Collections
PCM_OP_COLLECTIONS_GET_ACTION_HISTORY	Retrieving Collections Action History Information
PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS	Retrieving a List of Collections Actions
PCM_OP_COLLECTIONS_GET_BILLINFOS	Retrieving a List of Bill Units in Collections
PCM_OP_COLLECTIONS_GET_DUNNING_LETTER	Retrieving Dunning Letters
PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL	Retrieving Scenario Information
PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS	Getting All Valid Collections Scenarios Replacing a Collections Scenario
PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER	Adding Members to a Collections Group Creating a Collections Group
PCM_OP_COLLECTIONS_GROUP_CREATE	Creating a Collections Group
PCM_OP_COLLECTIONS_GROUP_DELETE	Deleting a Collections Group
PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER	Removing a Member from a Collections Group
PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO	Retrieving a Collections Group
PCM_OP_COLLECTIONS_GROUP_MODIFY	Modifying a Collections Group
PCM_OP_COLLECTIONS_GROUP_SET_PARENT	Changing the Parent of a Collections Group
PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY	Creating a System-Configured Promise-to-Pay Agreement Creating a Manually Configured Promise-to-Pay Agreement
PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES	Applying Finance Charges
PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES	Applying Late Fees
PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT	Assigning Bill Units Automatically Executing Automatic Collections Actions
PCM_OP_COLLECTIONS_POL_ASSIGN_DCA	Assigning Bill Units to a Debt Collections Agency
PCM_OP_COLLECTIONS_POL_CALC_DUE_DATE	Customizing Due Dates
PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION	Executing Pending Actions for a Bill Unit Performing Custom Collections Actions
PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO	Performing Custom Actions when a Bill Unit Leaves Collections Executing Automatic Collections Actions Exempting Bill Units from Collections
PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS	Customizing How to Apply Collections Actions to Collections Group Members

Table 8-1 (Cont.) Collections Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS	Getting All Valid Collections Scenarios
PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY	Breaking a Promise-to-Pay Agreement
PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT	Executing Pending Actions for a Bill Unit
PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY	Creating a System-Configured Promise-to-Pay Agreement
PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA	Customizing Dunning Letters Gathering and Storing Data for Dunning Letters
PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO	Executing Automatic Collections Actions
PCM_OP_COLLECTIONS_POL_SELECT_PROFILE	Configuring the Currency Used in Collections Mapping Bill Units to Collections Profiles Executing Automatic Collections Actions
PCM_OP_COLLECTIONS_PROCESS_BILLINFO	Assigning Bill Units Automatically Mapping Bill Units to Collections Profiles Performing Custom Actions when a Bill Unit Leaves Collections Executing Automatic Collections Actions Preparing Invoice Reminders Gathering and Storing Data for Dunning Letters Executing Pending Actions for a Bill Unit
PCM_OP_COLLECTIONS_REPLACE_SCENARIO	Replacing a Collections Scenario
PCM_OP_COLLECTIONS_RESCHEDULE_ACTION	Rescheduling an Action Scheduled for a Bill Unit Rescheduling a Promise-to-Pay Action
PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY	Breaking a Promise-to-Pay Agreement Canceling a Promise-to-Pay Agreement
PCM_OP_COLLECTIONS_SET_ACTION_STATUS	Changing the Status of a Collections Action Changing the Status of a Promise-to-Pay Action
PCM_OP_COLLECTIONS_SET_DUNNING_LETTER	Customizing Dunning Letters Gathering and Storing Data for Dunning Letters
PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER	Preparing Invoice Reminders
PCM_OP_COLLECTIONS_TAKE_ACTION	Applying Finance Charges Applying Late Fees Customizing How to Apply Collections Actions to Collections Group Members Performing Custom Collections Actions Executing Automatic Collections Actions Gathering and Storing Data for Dunning Letters Executing Pending Actions for a Bill Unit
PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS	Updating Amount and Payment Details
PCM_OP_GROUP_ADD_MEMBER	Adding Members to a Collections Group Creating a Collections Group

Table 8-1 (Cont.) Collections Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_GROUP_CREATE_GROUP	Creating a Collections Group
PCM_OP_GROUP_DELETE_GROUP	Deleting a Collections Group
PCM_OP_GROUP_SET_PARENT	Changing the Parent of a Collections Group
PCM_OP_INV_FORMAT_INVOICE	Retrieving Dunning Letters
PCM_OP_PYMT_POL_EXEC_COLLECTIONS_ACTION	Completing Promise-to-Pay Installments during Payment Processing
PCM_OP_UMS_SET_MESSAGE	Preparing Invoice Reminders

Customizing Collections Manager

See the following topics:

- Customizing Due Dates
- Customizing Dunning Letters
- Applying Finance Charges
- Applying Late Fees
- Assigning Bill Units Automatically
- Assigning Bill Units to a Debt Collections Agency
- Customizing How to Apply Collections Actions to Collections Group Members
- Performing Custom Collections Actions
- Performing Custom Actions when a Bill Unit Leaves Collections

Configuring the Currency Used in Collections

The default profile uses US dollars for the currency. To use a different currency for collections, edit the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode.

If a scenario is configured in any other currency, you must customize the PCM_OP_COLLECTIONS_POL_SELECT_PROFILE policy opcode to consider this scenario for collections.

Customizing Due Dates

Use PCM_OP_COLLECTIONS_POL_CALC_DUE_DATE to customize how collections actions due dates are determined. For example, for collections actions that fall on a holiday, you can customize this opcode to set the action due date to the following day.

By default, if any collections action falls on a Saturday or Sunday, this opcode sets the action due date to the following Monday.

Customizing Dunning Letters

PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA enables customization of dunning letter data before it is stored in the database. You can modify the type of data gathered by

customizing the opcode. For example, you can enrich the standard data with additional information, such as the date on which the account is inactivated.

By default, PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA gathers the following data for your dunning letters:

- Current overdue amount
- Currency type
- Current bill unit overdue date
- Account POID

> **Note:**
>
> By default, PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA gathers the data required for typical dunning letter templates, such as for the samples in *BRM_home*/**sys/data/config/stylesheets**. If you use custom templates with placeholders for nondefault data, you must customize this opcode.

PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA is called by PCM_OP_COLLECTIONS_SET_DUNNING_LETTER.

Applying Finance Charges

Use PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES to apply finance charges to overdue amounts.

This opcode is called by PCM_OP_COLLECTIONS_TAKE_ACTION to calculate the finance charge when the scenario associated with the bill unit calls for finance charges.

By default, its calculation is based on the finance charge action definition entered in Collections Configuration. The finance charge percentage from Collections Configuration is stored in the **/collections_action/finance_charge** object, which is created when the account bill unit enters a scenario that includes a finance charge action.

Required inputs include the POID of the **/collections_action/finance_charge** object for this finance charge, POID of the bill unit, POID of the account with which this bill unit is associated, overdue amount, and overdue date.

You can customize PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES to change the way a finance charge is calculated or to add functionality. For example, you can customize this opcode to calculate the finance charge from the customer's average daily balance rather than the current balance.

You can also customize PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES to find a specific balance group where finance charges apply. By default, BRM applies finance charges to the bill unit's default balance group.

By default, errors in the syntax of the input flist cause PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES to fail. Customizing this opcode may introduce additional error conditions.

By default, PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES returns the POID of the **/collections_action/finance_charge** object, the POID of the bill unit, the action status,

and the PIN_FLD_EVENT substruct. After customization, the return value depends on the features implemented.

If PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES fails, it returns an error in the error buffer. The transaction is rolled back.

Applying Late Fees

Use PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES to apply a late fee to overdue charges.

This opcode is called by PCM_OP_COLLECTIONS_CONFIG_SET_ACTION and PCM_OP_COLLECTIONS_TAKE_ACTION when the scenario associated with a bill unit calls for late fees. By default, it uses the late fee action definition entered in Collections Configuration. The late fee amount or percentage is stored in the **/collections_action/late_fee** object, which is created when the account bill unit enters a scenario that includes a late fee action.

Required inputs include the POID of the **/collections_action/late_fee** object for this late fee, the POID of the bill unit or the POID of the account to which the bill unit belongs, the overdue amount, and the overdue date.

PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES performs the following tasks by default:

- Gathers the late fee definition information and, if it is based on a percentage, calculates the amount.
- Converts currency, if necessary.

You can customize this opcode to change how the late fee is calculated or to add functionality. For example, you can customize this opcode to calculate a percentage-based late fee from the customer's average daily balance rather than from the current balance.

You can also customize PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES to find specific balance groups where late fees apply. By default, BRM applies late fees to the default balance group of the bill unit.

By default, errors in the input flist cause this opcode to fail. Customizing this opcode may introduce additional error conditions.

By default, this opcode returns the POID of the **/collections_action/late_fee** object and the status of the action. After customization, the return value depends on the features implemented.

If PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES fails, it returns an error in the error buffer. The transaction is rolled back.

Assigning Bill Units Automatically

Use PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT to automatically assign bill units to collections agents. This opcode is an empty hook.

This opcode can be modified to customize how account bill units are assigned to collections agents.

This opcode is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Assigning Bill Units to a Debt Collections Agency

Use PCM_OP_COLLECTIONS_POL_ASSIGN_DCA to automate the logic of selecting a debt collections agent (DCA) when multiple DCAs are configured. This opcode is an empty hook.

This opcode is called when a system collections action of type **Refer to outside agency** is run.

Customizing How to Apply Collections Actions to Collections Group Members

Use PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS to override the action target setting in the **/config/collections_actions** object. This opcode is an empty hook.

You can customize PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS to use additional attributes for deciding which bill units to apply a collections action to:

- The individual bill unit only
- The parent and all child bill units in the collections group
- All child bill units in the collections group

PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS is called by PCM_OP_COLLECTIONS_TAKE_ACTION.

You can customize this opcode to use additional attributes for deciding to which bill units to apply the collections action.

PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS must return in the output flist the **/account** POID and one of the following:

- The PIN_FLD_BILLINFO array with a list of **/billinfo** objects to which to apply the collections action.
- The PIN_FLD_TARGET field set to one of the following:
  - **0** to apply the collections action to the bill unit passed in the input flist.
  - **1** to apply the collections action to the parent and all child bill units in the collections group.
  - **2** to apply the collections action to all child bill units in the collections group.

Mapping Bill Units to Collections Profiles

Use PCM_OP_COLLECTIONS_POL_SELECT_PROFILE to map a bill unit to a collections profile. By default, this opcode maps all bill units to the default collections profile, but you can customize it to map bill units to custom profiles based on specific criteria.

For example, you can assign bill units to profiles based on credit score. You can map bill units with low credit scores to profiles with more aggressive collections scenarios and bill units with high credit scores to profiles with more lenient collections scenarios.

The default profile uses US Dollars for the currency. To use a different currency for collections, you must edit this opcode.

Your custom code must return the POID of the **/config/collections/profile** object or the collections profile to which the bill units map. BRM creates profile objects when you define one in Collections Configuration.

PCM_OP_COLLECTIONS_POL_SELECT_PROFILE is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Performing Custom Collections Actions

Use PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION to perform custom collections actions, such as sending SMS text messages to a customer's wireless phone. By default, this opcode sets the collections action status to **Pending** and then returns to the calling opcode.

This opcode is called by PCM_OP_COLLECTIONS_TAKE_ACTION.

The names and descriptions of custom actions are created in Collections Configuration and are stored in **/collections_action** objects. This information is then passed in an input flist to PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION. When you customize this opcode, you use this bill unit information to find any other information required for a particular custom action.

Performing Custom Actions when a Bill Unit Leaves Collections

Use PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO to perform custom tasks, such as cleaning up files or modifying a customer's credit score, when a bill unit exits collections. This opcode is an empty hook.

This opcode is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Managing Overdue Balance Collection

See the following topics:

- Creating or Updating Collections Actions
- Creating or Updating Collections Profiles
- Getting All Currently Defined Collections Actions.
- Getting All Currently Defined Collections Profiles
- Getting All Currently Defined Collections Scenarios
- Getting Details of a Collections Scenario
- Getting a List of Message Templates
- Deleting an Existing Collections Action
- Deleting an Existing Collections Profile
- Deleting an Existing Collections Scenario

Creating or Updating Collections Actions

Use PCM_OP_COLLECTIONS_CONFIG_SET_ACTION to create or update a collections action.

Collections Configuration calls this opcode when a user creates a new action or modifies an existing action.

As input, PCM_OP_COLLECTIONS_CONFIG_SET_ACTION takes the POID of the **/config/collections/action** object to create or update and it takes the action's name and type. This opcode uses the POID ID to determine whether to create a new action object or update an existing action object:

- If the PIN_FLD_POID field is set to **-1**, this opcode creates a new **/config/collections/action** object.
- If the PIN_FLD_POID field is set to the POID of an existing action object, this opcode updates the object.

PCM_OP_COLLECTIONS_CONFIG_SET_ACTION stops processing if the POID of the **/config/collections/action** object in the input flist is missing or invalid.

If successful, PCM_OP_COLLECTIONS_CONFIG_SET_ACTION returns the POID of the **/config/collections/action** object that was created or updated.

Creating or Updating Collections Profiles

Use PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE to create or update a collections profile.

Collections Configuration calls this opcode when a user creates or modifies a profile.

As input, PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE takes the POID of the **/config/collections/profile** object to create or update and it takes the profile's name, description, and the currency used for the profile. This opcode uses the POID ID to determine whether to create a new profile object or update an existing profile object:

- If the PIN_FLD_POID field is set to **-1**, this opcode creates a new **/config/collections/profile** object.
- If the PIN_FLD_POID field is set to the POID of an existing profile object, this opcode updates the object.

PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE stops processing if the POID of the **/config/collections/profile** object in the input flist is missing or invalid.

If PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE fails, it returns an error in the error buffer.

If successful, PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE returns the POID of the **/config/collections/profile** object that was created or updated.

Getting All Currently Defined Collections Actions

Use PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS to get a list of all currently defined collections actions.

Collections Configuration calls this opcode to retrieve a list of actions and their definitions.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/action** object that contains definitions of collections actions. The object is created when new actions are defined in Collections Configuration.

If PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID, name, description, and type of the actions returned. It also returns the PIN_FLD_OPCODE field, which shows the

opcode to use to run the actions. If any action involves sending a dunning letter, the output flist returns the POID of the template used by this action.

Getting All Currently Defined Collections Profiles

Use PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES to retrieve a list of currently defined collections profiles.

Collections Configuration calls this opcode to display all currently defined profiles.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/profile** object. This object is created when a new collections profile is defined in Collections Configuration.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID, name, description, and the currency used for each profile found.

Getting All Currently Defined Collections Scenarios

Use PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS to get a list of all collections scenarios and associated profiles.

Collections Configuration calls this opcode to list all currently defined scenarios and profiles.

As input, this opcode takes a dummy POID used to get the database ID. This opcode then searches the database for the **/config/collections/scenario** object. This object is created when a new collections scenario is defined in Collections Configuration.

This opcode also retrieves the **/config/collections/profile** object, which contains definitions of profiles that are associated with scenarios.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns a results array that contains the POID and name of each scenario. It also returns the POID and name of the profile associated with each scenario.

Getting Details of a Collections Scenario

Use PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL to get details of a particular collections scenario.

Collections Configuration calls this opcode to display details of the selected scenario.

As input, this opcode takes the POID of the **/config/collections/scenario** object.

This opcode stops processing if the POID of the **/config/collections/scenario** object in the input flist is missing or invalid.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL returns a results array that contains the POID of the **/config/collections/scenario** object, the name of the scenario, and the POID of the profile associated with the scenario.

It also returns an array that contains details for the action associated with the scenario, including the action object POID, name, description, type, and the PIN_FLD_OPCODE field,

which shows the opcode to use to run the action. If the action involves sending a dunning letter, the output flist returns the POID of the template used by this action.

Getting a List of Message Templates

Use PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES to get a list of templates.

When a user creates or updates an action that requires a template, Collections Configuration calls this opcode to display a list of available templates.

As input, PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES takes the action type, the template's location, and a dummy POID used to get the database ID.

The opcode performs the following:

- Reads the action type from the input flist. The PIN_FLD_ACTION_TYPE field specifies DUNNING_LETTER.
- Reads the **/config/invoice_templates/dunning** objects to retrieve dunning letter templates. BRM creates these objects and automatically creates subclasses of **/config/invoice_templates** when loading the dunning letter templates.

If this opcode fails, it returns an error in the error buffer.

If successful, this opcode returns an array containing the POID and name of each template that matches the locale and action type specified in the input flist.

Deleting an Existing Collections Action

Use PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION to delete an existing collections action.

Collections Configuration calls this opcode when a user deletes a collections action.

This opcode takes the POID of the **/config/collections/action** object to delete, verifies that the object is not used by any scenario, and deletes the object.

PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION stops processing if:

- The POID of the **/config/collections/action** object in the input flist is missing or invalid.
- The action object is used by a scenario.

If this opcode is not successful, it sets the PIN_FLD_RESULT field to **Fail** in the output flist. If the scenario is in use, an appropriate message is displayed in Collections Configuration.

If successful, this opcode returns the POID of the **/config/collections/action** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Deleting an Existing Collections Profile

Use PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE to delete an existing collections profile.

> **Note:**
>
> Users cannot delete a default profile created during the installation of Collections Configuration.

Collections Configuration calls this opcode when a user deletes a collections profile.

This opcode takes the POID of the **/config/collections/profile** object to delete, verifies that the object is not used by any scenario, and deletes the object.

PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE stops processing if:

- The POID of the **/config/collections/profile** object in the input flist is missing or invalid.
- The object is used by a scenario.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/config/collections/profile** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Deleting an Existing Collections Scenario

Use PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO to delete an existing collections scenario.

Collections Configuration calls this opcode when a user deletes a scenario.

This opcode takes the POID of the **/config/collections/scenario** object to delete, verifies that the object is not used by any bill units that are currently in collections and was not used by any bill units that were previously in collections, and deletes the object.

PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO stops processing if:

- The POID of the **/config/collections/scenario** object in the input flist is missing or invalid.
- The object is currently used by one or more bill units in collections.
- The object was previously used by one or more bill units that entered into collections. The collections scenario cannot be deleted even after a bill unit has exited from collections because the association between the collections scenario and the collections objects still remains.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/config/collections/scenario** object and the PIN_FLD_RESULT field, which shows the results of the deletion.

Managing Promise-to-Pay Agreements

To manage your customer's promise-to-pay agreements, see the following:

- Creating a System-Configured Promise-to-Pay Agreement
- Creating a Manually Configured Promise-to-Pay Agreement
- Updating Amount and Payment Details
- Canceling a Promise-to-Pay Agreement
- Breaking a Promise-to-Pay Agreement
- Rescheduling a Promise-to-Pay Action
- Changing the Status of a Promise-to-Pay Action
- Completing Promise-to-Pay Installments during Payment Processing

The promise-to-pay opcodes are called by Billing Care. You can customize your client application to call these opcodes for managing promise-to-pay agreements.

For more information about promise-to-pay agreements, see "Managing Promise-to-Pay Agreements" in *BRM Collections Manager*.

Creating a System-Configured Promise-to-Pay Agreement

Use PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY to create a promise-to-pay agreement with system-configured details. With system-configured agreements, Collections Manager calculates the installment schedule and installment amount for you. For example, a $500 promise-to-pay agreement could be created with five installments of $100 each that is due every 30 days.

To create a promise-to-pay agreement with system-configured details, pass in the following information in the opcode's input flist:

- The account and bill unit in collections (PIN_FLD_POID and PIN_FLD_BILLINFO_OBJ)
- The collections scenario (PIN_FLD_SCENARIO_OBJ)
- The total amount that the customer promises to pay (PIN_FLD_AMOUNT under the PIN_FLD_PROMISE_TO_PAY_INFO substruct)
- The due date for the first promise-to-pay installment (PIN_FLD_INVOKE_T under the PIN_FLD_PROMISE_TO_PAY_INFO substruct)
- The installment schedule and amount. Include the following fields under the PIN_FLD_PROMISE_TO_PAY_INFO substruct:
  - Either PIN_FLD_AMOUNT set to the amount owed for each installment, or PIN_FLD_NUM_MILESTONES set to the number of installment payments.

    If the installment amount is provided, Collections Manager calculates the number of installments as the total promise-to-pay amount divided by the installment amount. If the number of installments is provided, Collections Manager calculates the installment amount as the total promise-to-pay amount divided by the number of installments.
  - Either PIN_FLD_MILESTONE_INTERVAL set to the number of days between each installment payment, or PIN_FLD_DAYS set to the total number of days in which the full outstanding amount is paid off.

    If the interval is provided, Collections Manager calculates the agreement's total number of days by multiplying the interval by the number of installments. If the agreement's total number of days is provided, Collections Manager calculates the interval as the total number of days divided by the number of installments.

PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY performs the following actions:

1. Calls the PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY policy opcode to perform additional validation, such as preventing promise-to-pay agreements that go beyond 30 days. By default, this opcode is an empty hook.
2. Calculates the installment schedule and installment amount.
3. Creates a **/collections_action/promise_to_pay** object for each promise-to-pay installment.
4. Creates an **/event/activity/collections/promise_to_pay** object to record the details about the promise-to-pay agreement.
5. Reschedules all outstanding collections actions in the bill unit to continue, starting one day after the last promise-to-pay installment due date.

6. Generates an **/event/audit/collections/actions** audit event to record the changes to the collections actions.

Creating a Manually Configured Promise-to-Pay Agreement

Use PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY to create a promise-to-pay agreement with manually configured details. Manually configured agreements allow you to create installments with varying amounts owed and varying intervals between installments. For example, you could create a $500 promise-to-pay agreement with the installments shown in Table 8-2.

Table 8-2 Sample Manually Configured Promise-to-Pay Installments

Installment Number	Interval Between Installments	Amount Owed
1	0	$250
2	30 Days	$100
3	15 Days	$100
4	20 Days	$50

To create a promise-to-pay agreement with manually configured details, pass in the following information in the opcode's input flist:

- The account and bill unit in collections (PIN_FLD_POID and PIN_FLD_BILLINFO_OBJ)
- The collections scenario to apply (PIN_FLD_SCENARIO_OBJ)
- The promise-to-pay specification to apply (PIN_FLD_SPEC_NAME)
- The total amount that the customer promises to pay (PIN_FLD_AMOUNT under the PIN_FLD_PROMISE_TO_PAY_INFO substruct)
- The due date for the first promise-to-pay installment (PIN_FLD_INVOKE_T under the PIN_FLD_PROMISE_TO_PAY_INFO substruct)
- For each installment, its due date and the amount owed. The sum of all installments must equal 100% of the promise-to-pay agreement amount.

  For each installment, add a PIN_FLD_MILESTONES array with the following fields:

  – PIN_FLD_MILESTONE_INTERVAL set to the number of days between the previous installment and this installment.

    Set the interval for the first installment to **0**, because the due date of the first installment has already been set.

  – Either PIN_FLD_MILESTONE_AMOUNT set to the amount owed for this installment, or PIN_FLD_MILESTONE_PERCENTAGE set to the percentage of the total promise-to-pay amount that will be paid off in this installment.

When the promise-to-pay details are configured manually, PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY performs the following actions:

1. Validates that the promise-to-pay agreement's details adhere to the limits in the promise-to-pay specification (**/config/collections/promise_to_pay_spec** object).
2. Calls the PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY policy opcode to perform additional validation, such as preventing promise-to-pay agreements that go beyond 30 days. By default, this opcode is an empty hook.

3. Determines whether to automatically increase the customer's credit limit during the promise-to-pay period by reading the value of the PIN_FLD_CREDIT_LIMIT_FLAG field in the **/config/collections/promise_to_pay_spec** object:
   - If the field is set to **0**, the credit limit is not adjusted.
   - If the field is set to **1**, it calls the PCM_OP_BILL_SET_LIMIT_AND_CR opcode to temporarily increase the bill unit's credit limit, if needed. See "Managing Credit Limits and Sub-Balance Consumption Rules" for more information.
4. Creates a **/collections_action/promise_to_pay** object for each promise-to-pay installment.
5. Creates an **/event/activity/collections/promise_to_pay** object to record details about the promise-to-pay agreement.
6. Reschedules all outstanding collections actions in the bill unit to continue, beginning one day after the last promise-to-pay installment due date.
7. Generates an **/event/audit/collections/actions** audit event to record the changes to the collections actions.

Updating Amount and Payment Details

Use PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS to update the amount and payment details for a collections action.

Billing Care calls this opcode to update the amount and payment details for the **Promise to Pay** and **Collect Payment** actions.

If the input action type is **Promise to Pay** and the amount details are updated, the opcode updates the total outstanding amount in the **/collections_scenario** object with the difference amount.

If the input action type is **Promise to Pay** and the payment details are updated, depending on the option whether the updated details is applicable only to the current installment or includes the next installments, the opcode updates the details.

When calling this opcode, send only the data that needs to be updated. For example, in a scenario where only the amount details should be updated, if the payment details are also sent in the input flist with NULL values, the opcode updates the payment details of the current action with the NULL values.

Canceling a Promise-to-Pay Agreement

Use PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY to cancel a promise-to-pay agreement.

This opcode takes as input the POID of the **/event/activity/collections/promise_to_pay** object, the POID of the **/account** object, the POID of the **/collections_scenario** object, and the POID of the **/billinfo** object. Then, the opcode does the following:

1. Cancels all outstanding promise-to-pay installments and deletes all corresponding **/schedule** objects.
2. Sets the status of the overall promise-to-pay agreement to one of the following:
   - **Broken**: Set to broken only if this opcode is called by the PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY policy opcode.
   - **Canceled**: Set to canceled if this opcode is called by an application.

3. Reads the value of the PIN_FLD_CREDIT_LIMIT_FLAG field in the **/config/collections/promise_to_pay_spec** object:
   - If the value is **1**, calls PCM_OP_BILL_SET_LIMIT_AND_CR to decrease the bill unit's temporary credit limit. See "Managing Credit Limits and Sub-Balance Consumption Rules for more information".
   - If the value is **0**, the credit limit is not adjusted.
4. Reschedules all outstanding collections actions to start from the next day and updates the corresponding **/schedule** objects.
5. Removes the reference of the **/event/activity/collections/promise_to_pay** object from the **/collections_scenario** object.

Breaking a Promise-to-Pay Agreement

Use the PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY policy opcode to break a promise-to-pay agreement after a customer misses a promise-to-pay installment. This opcode is called by PCM_OP_COLLECTIONS_PYMT_COLLECT.

By default, this opcode does the following, but you can customize it to meet your business needs.

1. Sets the status of the current promise-to-pay installment to **Broken**.
2. Calls the PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY opcode to cancel all outstanding installments, set the status of the overall promise-to-pay agreement to **Broken**, and reschedule all collections actions to start from the following day. See "Canceling a Promise-to-Pay Agreement" for more information.

Rescheduling a Promise-to-Pay Action

Use PCM_OP_COLLECTIONS_RESCHEDULE_ACTION to reschedule a **Promise to Pay** action that was scheduled by a bill unit's collections scenario. See "Rescheduling an Action Scheduled for a Bill Unit".

Changing the Status of a Promise-to-Pay Action

Use PCM_OP_COLLECTIONS_SET_ACTION_STATUS to change the status of a **Promise to Pay** action. See "Changing the Status of a Collections Action".

Completing Promise-to-Pay Installments during Payment Processing

Use the PCM_OP_PYMT_POL_EXEC_COLLECTIONS_ACTION policy opcode to automatically complete a promise-to-pay installment as part of the payment process. This opcode is called by PCM_OP_PYMT_COLLECT.

This policy opcode determines whether a payment is for a promise-to-pay installment and then returns the PIN_FLD_RESULTS[0].PIN_FLD_RESULT output flist field set to one of the following:

- PIN_PYMT_RES_NO_EXEC_COLLECTIONS_ACTION (**1**): The payment is *not* for a promise-to-pay installment. This is the default.
- PIN_PYMT_RES_EXEC_COLLECTIONS_ACTION (**0**): The payment is for a promise-to-pay installment.

When PIN_PYMT_RES_EXEC_COLLECTIONS_ACTION (**0**) is returned, PCM_OP_PYMT_COLLECT changes the promise-to-pay installment's status to **Completed**.

By default, the PCM_OP_PYMT_POL_EXEC_COLLECTIONS_ACTION policy opcode returns PIN_PYMT_RES_EXEC_COLLECTIONS_ACTION (**0**) when the following is true, but you can customize the policy opcode to meet your business needs.

- The bill unit is in collections.
- In the PIN_FLD_PAYMENT_REASONS input flist array, the PIN_FLD_REASON_ID is set to **15** and the PIN_FLD_REASON_DOMAIN_ID field is set to **5**.

  For information about the reason ID values, see the *BRM_home***/sys/msgs/reasoncodes/reasons.en_US** file.

Managing Collections Groups

See the following topics:

- Creating a Collections Group
- Adding Members to a Collections Group
- Removing a Member from a Collections Group
- Deleting a Collections Group
- Retrieving a Collections Group
- Modifying a Collections Group
- Changing the Parent of a Collections Group

Creating a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_CREATE to group bill units into a collections group.

This opcode takes as input the **/account** POID and the **/billinfo** POID of the group owner, the **/billinfo** POIDs of the members of the group, and the name of the collections group.

If you pass a member's details in the input flist, the opcode calls PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER opcode to add the member to the collections group.

PCM_OP_COLLECTIONS_GROUP_CREATE performs the following actions:

- Validates that the parent bill unit is not already a parent or member of another collections group.
- Calls PCM_OP_GROUP_CREATE_GROUP to create the **/group/collections_targets** object. When a group is created, events are created.

  The **/event/group/collections_targets** event contains details of a parent and members in the PIN_FLD_MEMBERS array.

  The **/event/group/parent** event contains details of a parent.
- Calls PCM_OP_GROUP_ADD_MEMBER to add members to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, PCM_OP_COLLECTIONS_GROUP_CREATE returns the POID of the **/group/collections_targets** object.

Adding Members to a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER to add a child member to an existing collections group. This opcode checks whether the prospective member is an account that has a nonpaying bill unit. Such accounts cannot be added as members of a collections group. If the account is already in the collections group, it is ignored.

This opcode takes as input the POID of the **/group/collections_targets** object, the **/billinfo** POID of the new member, and the name of the collections group.

PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER performs the following actions:

- Validates that the new child member is not a parent or member of the existing collections group.
- Calls PCM_OP_GROUP_ADD_MEMBER to add the bill unit to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER returns the POID of the **/group/collections_targets** object.

Removing a Member from a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER to remove a member from an existing collections group.

This opcode takes as input the POID of the **/group/collections_targets** object and the POID of the **/billinfo** object to remove the member and returns the POID of the **/group/collections_targets** object.

Deleting a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_DELETE to delete an existing collections group.

This opcode takes as input the POID of the **/group/collections_targets** object to delete, calls PCM_OP_GROUP_DELETE_GROUP opcode to delete the specified object, and returns the POID of the **/group/collections_targets** object.

If the collections group is already in collections, the opcode throws an error.

Retrieving a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO to retrieve:

- Whether the specified bill unit is a member of a collections group.
- The pending receivables for each bill unit. If a bill unit is a parent of a collections group, it also provides the name of the collections group and its child member bill units.

This opcode is called directly by Billing Care.

PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO takes as input the **/account** or **/billinfo** POID and performs the following actions:

- If the **/billinfo** POID is not passed in, retrieves all **/billinfo** objects associated with the **/account** object.

- For each **/billinfo** object, reads the associated **/group/collections_targets** object to determine whether the bill unit belongs to a collections group and, if it does, whether it is a parent bill unit or a child bill unit.
  - If it is not a member of a collections group, returns the **/group/collections_targets** POID and the PIN_FLD_BOOLEAN field set to **0**.
  - If it is a child in a collections group, returns the **/group/collections_targets** POID and the PIN_FLD_BOOLEAN field set to **1**.
  - If it is a parent in a collections group, returns the **/group/collections_targets** POID, the PIN_FLD_BOOLEAN field set to **1**, and the pending receivables of all group members.

Modifying a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_MODIFY to modify an existing collections group, such as renaming the collections group, changing the parent member, and replacing all of the existing child members.

This opcode takes as input the POID of the **/group/collections_targets** object and the information to change, modifies the collections group, and returns the POID of the **/group/collections_targets** object.

Changing the Parent of a Collections Group

Use PCM_OP_COLLECTIONS_GROUP_SET_PARENT to change a collections group's existing parent bill unit.

This opcode takes as input the POID of the **/group/collections_targets** object, the name of the calling program, and the POID of the new parent bill unit.

PCM_OP_COLLECTIONS_GROUP_SET_PARENT performs the following actions:

- Validates that the new parent bill unit is not a child of the existing collections group.
- Validates that the new parent bill unit is not already a parent of another collections group.
- Calls PCM_OP_GROUP_SET_PARENT to assign the new parent bill unit to the **/group/collections_targets** object.

If the opcode fails, it returns an error in the error buffer. The transaction is rolled back.

If successful, PCM_OP_COLLECTIONS_GROUP_SET_PARENT returns the POID of the **/group/collections_targets** object.

Performing Manual Collections Actions

To enable your custom client application to update Collections Manager after manual actions occur, customize it to accept the information required by the target opcode's input flist and pass the information to the target opcode.

- Assigning Bill Units to a Collections Agent
- Adding Actions to a Collections Scenario
- Exempting Bill Units from Collections
- Rescheduling an Action Scheduled for a Bill Unit
- Changing the Status of a Collections Action

- Replacing a Collections Scenario

Assigning Bill Units to a Collections Agent

Use PCM_OP_COLLECTIONS_ASSIGN_AGENT to assign bill units to a collections agent.

This opcode is called when a collections manager assigns a bill unit to a particular agent.

This opcode takes as input the POID of the agent's account and the PIN_FLD_BILLINFO array, which specifies the bill units to assign to the agent.

This opcode performs the following actions:

- Assigns each bill unit in the PIN_FLD_BILLINFO array to the agent.
- Updates bill unit scenarios with the agent's account POID.
- Returns the account POID of the agent responsible for the bill units and an array of the bill units assigned to the specified agent.

This opcode stops processing if the bill unit information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the account POID of the agent responsible for the bill units and an array of the bill units assigned to the agent.

Adding Actions to a Collections Scenario

Use PCM_OP_COLLECTIONS_ADD_ACTION to add collections actions to a bill unit's collections scenario.

This opcode is called when CSRs add actions to a collections scenario.

This opcode takes as input a dummy POID, the **/collections_scenario** POID, the CSR's login account POID, the new action object's POID, and the action's due date.

This opcode performs the following actions:

- Checks the CSR account POID to make sure the user is authorized to add actions.
- Adds a new action to a scenario and updates the output flist with the new action POID.
- Sets the new action's status. If collections action dependencies are enabled, checks whether the subsequent action's status is set to **Pending**. If so, sets the new action's status to **Pending** and updates the subsequent action's status to **Waiting For Dependents**. If not, sets the new action's status to **Waiting For Dependents**.

  > **Note:**
  >
  > Actions cannot be inserted before, between, or on the same day as any canceled or completed actions.

- Checks to see if the PIN_FLD_DAYS field is present in the input flist. If so, PCM_OP_COLLECTIONS_ADD_ACTION checks the contents of the field to determine whether the actions that follow must be rescheduled.
  - If PIN_FLD_DAYS is equal to **0**, the actions are not rescheduled.

  - If PIN_FLD_DAYS is less than **0**, the actions are canceled.
  - If PIN_FLD_DAYS is greater than **0**, the value specified indicates the number of days the actions is delayed.
- Checks to see if the type of collections action is **Collect Payment** and the Credit Card or Debit Card credentials are specified and does the following:
  - Creates a **/payinfo** object with the credentials in the input flist.
  - Creates the **/collections_action/collect_payment** action to hold the amount to be paid and a link to **/payinfo** object.
- Checks to see if the type of collections action is **Promise to Pay** and does the following:
  - Validates if the promise-to-pay agreement is already called for the bill unit and the due date is not before INVOKE_T of the active promise.
  - Updates the outstanding amount under the **/collections_scenario** object with the amount for the new payment milestone.
- Updates the output flist with the new data for each rescheduled action.

PCM_OP_COLLECTIONS_ADD_ACTION stops processing if:

- The CSR account POID does not match the POID of a user who has permission to add actions.
- The action information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns PIN_FLD_POID set to the POID of the new action added and updates the output flist with the new data for each rescheduled action.

Exempting Bill Units from Collections

Use PCM_OP_COLLECTIONS_EXEMPT_BILLINFO to exempt bill units from collections.

This opcode is called when a CSR exempts a bill unit from collections.

This opcode takes as input the POID of the bill unit to exempt from collections and the POID of the account that owns the bill unit.

If the bill unit is already in collections, this opcode performs the following actions:

- Calls PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO to remove the bill unit from the scenario.
- Cancels all pending and error actions scheduled for this bill unit.
- Sets the bill unit's PIN_FLD_SCENARIO_OBJ field to NULL to indicate that this bill unit is not in collections.
- Sets the exemption flag so that the bill unit never enters collections, even if it has an overdue balance.

If the bill unit is not in collections, PCM_OP_COLLECTIONS_EXEMPT_BILLINFO sets the exemption flag so that the bill unit never enters collections, even if it has an overdue balance.

This opcode stops processing if the input flist does not include a valid bill unit POID.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the exempted bill unit and the POID of the account to which the bill unit belongs.

Rescheduling an Action Scheduled for a Bill Unit

Use PCM_OP_COLLECTIONS_RESCHEDULE_ACTION to reschedule an action that was scheduled by a bill unit's collections scenario.

This opcode is called when a CSR reschedules a particular action to be performed on a bill unit. For example, if a customer promises payment by a certain day, the CSR can reschedule the inactivation of the bill unit.

This opcode takes as input the POID of the action to be rescheduled, the account POID of the CSR that is rescheduling the action, and the new due date for the action.

PCM_OP_COLLECTIONS_RESCHEDULE_ACTION performs the following actions:

- Checks the account POID to verify that the CSR is authorized to reschedule the action. If the account POID matches the account of the agent who originally scheduled the action or if it is the account of a manager, the opcode allows rescheduling.
- Reads the PIN_FLD_DUE_T input field to obtain a new due date for the action.
- Checks to see if the PIN_FLD_DAYS field is present in the input flist. If so, this field determines whether the actions that follow are postponed.
  - If PIN_FLD_DAYS is equal to **0**, the actions are not postponed.
  - If PIN_FLD_DAYS is less than **0**, the actions are canceled.
  - If PIN_FLD_DAYS is greater than **0**, the value indicates the number of days to postpone the actions.
- Updates the output flist with the new data for each action.

PCM_OP_COLLECTIONS_RESCHEDULE_ACTION stops processing if:

- The CSR's account POID does not match the POID of an authorized user.
- The action information in the input flist is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the POID of the rescheduled action and updates the output flist with the new data for each action affected by the rescheduling.

Changing the Status of a Collections Action

Use PCM_OP_COLLECTIONS_SET_ACTION_STATUS to change the status of a manual collections action.

This opcode is called when a CSR updates the status of a manual action. For example, after finishing a phone call, an agent must update the action's status to **Completed**.

This opcode takes as input the POID of the action object and the new status for the action.

If collections action dependencies are enabled, PCM_OP_COLLECTIONS_SET_ACTION_STATUS performs the following actions:

- Checks whether the action's current status is set to **Pending**. If so, it changes the current action's status and then updates the subsequent action's status from **Waiting For Dependents** to **Pending**. If not, it generates an error.

- If the action's status is changed to **Cancelled**, checks the value of the PIN_FLD_FLAGS flist field and the **/collections_action** object's PIN_FLD_ACTION_MODE field:
  - If PIN_FLD_ACTION_MODE is set to **0**, the action is mandatory. The action cannot be canceled. The opcode generates an error.
  - If PIN_FLD_ACTION_MODE is set to **1** and PIN_FLD_FLAGS is set to **0**, the opcode changes the action's status to **Cancelled**.
  - If PIN_FLD_ACTION_MODE is set to **1** and PIN_FLD_FLAGS is set to **1**, the opcode changes the action's status to **Cancelled** and the status of all actions that follow to **Cancelled**.
- If the action's status is changed to **Completed**, checks the status of the PIN_FLD_FLAGS flist field:
  - If it is set to **0**, the opcode changes the action's status to **Completed** and reschedules the due dates of all outstanding actions in the scenario.
  - If it is set to **2**, the opcode changes the action's status to **Completed** and leaves the due dates of all outstanding actions unchanged.

If collections action dependences are disabled:

- Changes the current action's status.
- If the action status is changed to **Cancelled**, checks the value of the PIN_FLD_FLAGS flist field:
  - If it is set to **0**, the opcode changes the action's status to **Cancelled**.
  - If it is set to **1**, the opcode changes the action's status to **Cancelled** and the status of all actions that follow to **Cancelled**.

PCM_OP_COLLECTIONS_SET_ACTION_STATUS stops processing if the POID of an action is missing or invalid.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back.

If successful, this opcode returns the POID of the action whose status was changed.

Replacing a Collections Scenario

Use PCM_OP_COLLECTIONS_REPLACE_SCENARIO to replace the existing collections scenario for a bill unit with a new collections scenario. You can replace the scenario of a bill unit that is already in collections.

Collections Configuration calls this opcode to replace the scenario of a bill unit that is already in collections.

As input, this opcode takes the POID of the bill unit already in collections.

This opcode creates the **/event/activity/collections/replace_scenario** activity event object to store the bill unit details such as the bill unit POID, the current collections scenario POID, and the new collections scenario POID. If the new collections scenario object is not specified, this opcode calls PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS to get a list of all the valid collections scenarios applicable for the bill unit.

After the new scenario is identified, PCM_OP_COLLECTIONS_REPLACE_SCENARIO exits out of the existing collections scenario, cancels all the pending actions, deletes the scheduled objects for the pending actions, and updates the bill unit with the POID of the new collections scenario.

Performing System Collections Actions

Collections Manager performs a variety of actions on bill units with overdue balances. Some actions happen automatically, and others require manual intervention.

Automatic actions are performed by the system. They include calculating overdue balances, applying late fees, and updating bill unit balances.

Collections Manager uses the following opcodes to perform system actions on the bill units in collections:

- Executing Automatic Collections Actions
- Preparing Invoice Reminders
- Gathering and Storing Data for Dunning Letters
- Retrieving Dunning Letters
- Executing Pending Actions for a Bill Unit

Executing Automatic Collections Actions

Use PCM_OP_COLLECTIONS_PROCESS_BILLINFO to do the following:

- Determine whether an account's bill unit should enter or exit the collections process.
- Perform collections actions, such as sending notices to customers or applying late charges.

The **pin_collections_process** utility calls this opcode for each bill unit with an overdue balance that meets a minimum amount. This opcode then processes each bill unit to evaluate its collections status and perform any necessary collections actions.

PCM_OP_COLLECTIONS_PROCESS_BILLINFO calls PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO. You can use PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO to include additional criteria to determine whether bill units should enter and exit collections. Based on the status flag in the input flist, this opcode identifies if it is being called during entry or exit. PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO is an empty hook.

> **Tip:**
>
> The source code for PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFOis not shipped with BRM.

When PCM_OP_COLLECTIONS_PROCESS_BILLINFO receives a bill unit, it performs the following actions:

- Calculates the exact overdue balance and days late for the bill unit.
- Checks the bill unit's current collections status.

  If the bill unit is *not* in collections, PCM_OP_COLLECTIONS_PROCESS_BILLINFO:

  – Calls PCM_OP_COLLECTIONS_POL_SELECT_PROFILE to assign the bill unit to a collections profile. See "Mapping Bill Units to Collections Profiles".

  – Maps the collections profile to a collections scenario.

  - Creates a **/collections_scenario** object and a **/collections_action** object for each collections activity that needs to be performed.
  - Sets the status of all actions in the collections scenario. If collections action dependencies are enabled, sets the first action's status to **Pending** and all subsequent actions' status to **Waiting For Dependents**. If multiple actions are scheduled for the first due date, sets the status of those actions to **Pending** and sets the status of actions scheduled for later due dates to **Waiting For Dependents**.

    If collections action dependencies are disabled, sets the status of all actions to **Pending**.
  - Generates the **/event/audit/collections/action** notification event.
  - Calls PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT to assign a collections agent to the bill unit. See "Assigning Bill Units Automatically".

  If the bill unit *is already* in collections, PCM_OP_COLLECTIONS_PROCESS_BILLINFO sets the bill unit's action status to REMAIN_IN and updates the bill unit's overdue balance and days late.

- Determines whether the bill unit meets the exit criteria:

  If it *meets* the exit criteria, PCM_OP_COLLECTIONS_PROCESS_BILLINFO changes the bill unit's action status to **Completed**, generates the **/event/audit/collections/action** notification event, and calls PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO to perform any custom actions that you specify. (See "Performing Custom Actions when a Bill Unit Leaves Collections".) Processing is complete.

  If it *does not meet* the exit criteria, PCM_OP_COLLECTIONS_PROCESS_BILLINFO:

  - Finds all of the bill unit's actions that are set to **Pending**.
  - If a *manual* action is required, generates the **/event/audit/collections/action** notification event. Processing is complete.
  - If a *system or custom* action is required, calls PCM_OP_COLLECTIONS_TAKE_ACTION to run the action. See "Executing Pending Actions for a Bill Unit".

- Evaluates the bill unit to determine if it now meets the exit criteria for the scenario:

  If the bill unit *does not* meet the exit criteria, processing ends for this bill unit.

  If the bill unit *does* meet the exit criteria, PCM_OP_COLLECTIONS_PROCESS_BILLINFO:

  - Changes the bill unit's action status to **Completed**.
  - Calls PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO to perform any custom exit actions that you specify. See "Performing Custom Actions when a Bill Unit Leaves Collections".
  - Generates the **/event/audit/collections/action** notification event.

Preparing Invoice Reminders

Use PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER to prepare a reminder message for customers with overdue payments. This message is later added to your invoices or custom documents via the Universal Message Store (UMS) framework. See "Using BRM Messaging Services" in *BRM Developer's Guide*.

This opcode is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO when a collections scenario calls for an invoice reminder.

This opcode performs the following actions:

- Calls PCM_OP_UMS_SET_MESSAGE to create a **/message** object.
- Returns the POID of the **/collections_action/invoice_reminder** object and the status to PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Gathering and Storing Data for Dunning Letters

Use PCM_OP_COLLECTIONS_SET_DUNNING_LETTER to gather and store data for dunning letters.

This opcode is called by PCM_OP_COLLECTIONS_TAKE_ACTION when a collections scenario requires a dunning letter.

This opcode performs the following actions:

- Calls PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA to retrieve data for the dunning letter and to perform any customizations that you implement. See "Customizing Dunning Letters".
- Converts the data into XML format and saves it as a buffer field in the **/collections_action/dunning_letter** object for this letter.
- Returns the POID of the **/collections_action/dunning_letter** object and the status to PCM_OP_COLLECTIONS_PROCESS_BILLINFO.

Retrieving Dunning Letters

Use PCM_OP_COLLECTIONS_GET_DUNNING_LETTER to retrieve dunning letters from the BRM database.

This opcode is called directly by the **pin_collections_send_dunning** utility.

This opcode performs the following actions:

- Retrieves the **/collections_action/dunning_letter** object from the BRM database. This object contains the XML data to be included in the letter.
- Retrieves the **/config/invoice_templates/dunning** object from the BRM database. This object contains the XSLT style sheet used to format the dunning letter.

> **Note:**
>
> Dunning letter templates are stored as **/config/invoice_templates/dunning** objects. The **/config/invoice_templates** storable class is automatically subclassed when you load a dunning letter template.

- Calls PCM_OP_INV_FORMAT_INVOICE to create the final dunning letter.

Executing Pending Actions for a Bill Unit

Use PCM_OP_COLLECTIONS_TAKE_ACTION to run pending actions for a bill unit.

This opcode is called by either PCM_OP_COLLECTIONS_PROCESS_BILLINFO or the **pin_deferred_act** utility to run actions.

As input, this opcode takes the POID of the action to run, the POID of the bill unit to process, and the POID of the account that owns the bill unit. This opcode then determines the type of action that needs to be run:

- If it requires a *manual* action, this opcode stops processing.
- If it requires a *system* action, this opcode calls other opcodes to run the desired action:
  - Prepare a dunning letter. See "Gathering and Storing Data for Dunning Letters".
  - Apply finance charges. See "Applying Finance Charges".
  - Apply late fees. See "Applying Late Fees".
- If it requires a *custom* action, this opcode calls PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION to perform any custom actions that you specify. See "Performing Custom Collections Actions".

PCM_OP_COLLECTIONS_TAKE_ACTION calls PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT to performs auto-collect of the due amount for **promise_to_pay** and **collect_payment** actions, if the customer has opted to pay off the due amount through credit card or direct debit.

Then, this opcode sets the action's status to **Completed** and updates the bill unit's overdue date and overdue amount. This opcode generates an **/event/audit/collections/action** notification event.

If collections action dependencies are enabled, PCM_OP_COLLECTIONS_TAKE_ACTION also changes the subsequent action's status from **Waiting For Dependents** to **Pending** and reschedules the due dates of any outstanding collections actions in the scenario.

> **Note:**
>
> If multiple actions are scheduled for the same day and are all set to **Pending**, this opcode waits until all of those actions are completed or canceled before changing the subsequent action's status from **Waiting For Dependents** to **Pending**.

If PCM_OP_COLLECTIONS_TAKE_ACTION is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the **/collections_action** object for this action and the status of action.

Retrieving Collections Information

See the following topics:

- Retrieving a List of Bill Units in Collections
- Retrieving Aging Buckets Information
- Retrieving Scenario Information
- Getting All Valid Collections Scenarios
- Retrieving a List of Collections Actions
- Retrieving Collections Action History Information

Retrieving a List of Bill Units in Collections

Use PCM_OP_COLLECTIONS_GET_BILLINFOS to get a list of bill units that are in collections.

This opcode is called to display the bill units in collections that meet the criteria defined by a CSR.

The input flist must contain a dummy POID, which is used to get the database ID for the search. To retrieve bill units based on specific criteria, you can use the following optional input:

- The ID of the bill unit
- The status of the bill unit
- The payment type of the bill unit
- A date range; this retrieves bill units whose bills were generated between the specified dates
- The status of the account
- The account number (PIN_FLD_ACCOUNT_NO); this narrows the search to a particular account whose bill units are in collections
- The name of the company to which an account belongs
- The name of the customer on the account
- The account POID of the agent assigned to the bill units
- The name of the collections scenario assigned to the bill units
- The name of the collections profile associated with the bill units
- The overdue amount range
- The overdue days range; this narrows the search to bill units that are overdue for the specified number of days
- A flag indicating whether to get assigned bill units or unassigned bill units

If the opcode fails, a NULL flist is returned.

If successful, PCM_OP_COLLECTIONS_GET_BILLINFOS returns a list of bill units that meet the search criteria and details for the fields specified on the input flist.

> **Note:**
>
> This opcode uses step search when searching for bill units.

Retrieving Aging Buckets Information

Use PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS to retrieve a bill unit's aging buckets details.

This opcode is called when displaying the distribution of a bill unit's overdue balance over a number of aging buckets.

This opcode takes the POID of the bill unit and performs the following actions:

- Retrieves aging bucket information, which includes the total number of buckets and the number of overdue days per bucket from the **/config/collections/aging_buckets** object.
- Determines the exact number of overdue days per each aging bucket for the specified bill unit.
- Calculates the amount due in each bucket.

PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS stops processing if the bill unit information in the input flist is missing or invalid.

If this opcode is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the POID of the bill unit, an array of the aging buckets, the number of overdue days for each bucket, and the amount due in each bucket.

Retrieving Scenario Information

Use PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL to retrieve details about a bill unit's collections scenario.

This opcode is called to display the collections activity details that CSRs use to work with the collections actions that are scheduled for a bill unit.

This opcode takes the POID of the **/collections_scenario** object and performs the following actions:

- Reads the **/collections_scenario** object information.
- Retrieves a list of actions for the specified scenario.
- Retrieves history information for each action.

PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL stops processing if the input flist does not include a valid POID for a **/collections_scenario** object.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, this opcode returns the output flist that contains the POID of the bill unit, the POID of the scenario assigned to the bill unit, scenario details, and an array listing the POID and status of each scheduled action.

Getting All Valid Collections Scenarios

Use PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS to evaluate all the collections scenarios for the bill unit using the default entry criteria (**entry_amount** and **entry_days**), the severity attribute, and additional configurable parameters and list the valid scenarios for the bill unit.

This opcode is called to display the list of valid scenarios during scenario assignment or replacement.

This opcode takes the entry criteria (**entry_amount** and **entry_days**), the severity attribute, and additional configurable parameters and performs the following actions:

- Searches for collections scenarios matching the criteria.
- For each collections scenario, calls PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS to determine if a scenario is valid or not for the current bill unit.

PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS takes the scenario as input, reads the **/config/collections/scenario_params** object for the additional parameters, and validates whether the scenario is valid for the bill unit in collections. The valid collections scenario is added to the list of valid collections scenarios. If the collections scenario is not valid, the opcode evaluates the next collections scenario.

- Returns a list of all the valid scenarios. The results are ordered by the entry amount in the descending order and severity in the ascending order.

Retrieving a List of Collections Actions

Use PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS to retrieve a list of collections actions assigned to collections agents.

This opcode is called when collections managers request an overview of the workload for the collections agents they supervise.

The input flist contains the following:

- The agent's account POID. This returns a list of actions that are currently assigned to the specified agent. If you specify the account POID as type-only, this opcode retrieves the actions for all collections agents.
- PIN_FLD_START_T and PIN_FLD_END_T are optional. You specify one of these fields to limit the actions returned to those that are due either after the start time or before the end time.
- The PIN_FLD_THRESHOLD field specifies the number of actions to retrieve. The default is all actions.

PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS returns an array of details for the collections action. The return data includes the agent's name and account POID, the POID of the action object, the action's name and due date, and the opcode to run for the action.

> **Note:**
>
> This opcode uses step search when searching for bill units.

Retrieving Collections Action History Information

Use PCM_OP_COLLECTIONS_GET_ACTION_HISTORY to find past information about a particular collections action.

This opcode is called to display details about when an action was assigned to a collections agent, reassigned, rescheduled, and so on.

As input, this opcode takes the POID of the **/collections_action** object for which to retrieve historic data.

This opcode stops processing when the input flist does not include a valid POID for a **/collections_action** object.

If this opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure.

If successful, PCM_OP_COLLECTIONS_GET_ACTION_HISTORY returns the POID of the **/collections_action** object and a results array with history details for the collections action. The

array includes the action's status and due date, the assigned agent's name and account POID, the date when the last change occurred, and the description of the change.

9

Customer Management Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) customer management opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- About Account Locking
- Transaction Handling During Account Creation
- Creating Accounts
- Managing Name and Address Information
- Authenticating and Authorizing Customers
- Managing and Customizing Profiles
- Modifying an Account
- Creating Services
- Modifying Services
- Finding Accounts
- Deleting Accounts
- Customizing Customer Payment Information
- Setting Account, Service, and Bill Unit Status
- Getting Life Cycle States
- Amending Creditor Information
- Amending a Mandate
- Canceling a Mandate
- Managing Deferred Actions
- Managing Service Groups
- Managing Profile Sharing Groups
- Using Access Control Lists
- Customizing the Account Dump Utility (ADU)
- Business Profile Opcode Workflows
- Creating and Managing Account Hierarchies

Opcodes Described in This Chapter

Table 9-1 lists the opcodes described in this chapter.

Caution:

- Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
- Do not use SQL commands to change data in the database. Always use the API.

Table 9-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_FIND	Finding a Customer's Account Information Authenticating User Actions
PCM_OP_ACT_FIND_VERIFY	Authenticating User Actions Customizing Authentication Checks
PCM_OP_ACT_LOGIN	Customizing Authentication Checks
PCM_OP_ACT_POL_EVENT_LIMIT	Inactivating Accounts that Exceed a Specified Limit
PCM_OP_ACT_POL_EVENT_NOTIFY	Inactivating Accounts that Exceed a Specified Limit Scheduling Deferred Actions
PCM_OP_ACT_POL_SPEC_VERIFY	Authenticating User Actions Customizing Authentication Checks Enabling Duplicate Session Checking
PCM_OP_ACT_POL_VALIDATE_SCHEDULE	Performing Policy Checks before Scheduling Deferred Actions
PCM_OP_ACT_SCHEDULE_CREATE	Scheduling Deferred Actions
PCM_OP_ACT_SCHEDULE_DELETE	Deleting Deferred Actions
PCM_OP_ACT_SCHEDULE_EXECUTE	Executing Deferred Actions Scheduling Deferred Actions About Deferred Actions When Using PCM_OP_CUST_SET_STATUS
PCM_OP_ACT_SCHEDULE_MODIFY	Modifying Deferred Actions
PCM_OP_ACT_VERIFY	Implementing Password Encryption
PCM_OP_ADU_POL_DUMP	Customizing the Account Dump Utility (ADU)
PCM_OP_ADU_POL_VALIDATE	Customizing the Account Dump Utility (ADU)
PCM_OP_ADU_VALIDATE	Customizing the Account Dump Utility (ADU)
PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	Getting Life Cycle States
PCM_OP_BILL_GROUP_ADD_MEMBER	Adding Accounts to Account Hierarchies Moving Accounts from One Account Hierarchy to Another
PCM_OP_BILL_GROUP_CREATE	Creating Account Hierarchies
PCM_OP_BILL_GROUP_DELETE	Deleting Account Hierarchies
PCM_OP_BILL_GROUP_DELETE_MEMBER	Deleting Accounts from Account Hierarchies
PCM_OP_BILL_GROUP_GET_CHILDREN	Getting a List of Child Accounts in an Account Hierarchy
PCM_OP_BILL_GROUP_GET_PARENT	Finding the Parent of an Account Hierarchy
PCM_OP_BILL_GROUP_MOVE_MEMBER	Scheduling Deferred Actions

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_MAKE_BILL_NOW	Canceling a Service Group
PCM_OP_CUST_ACCTINFO	Customizing Automatic Account Creation (AAC) Information
PCM_OP_CUST_AMEND_CREDITOR_INFO	Amending Creditor Information Updating a Mandate
PCM_OP_CUST_AMEND_MANDATE	Amending a Mandate Updating a Mandate
PCM_OP_CUST_CANCEL_MANDATE	Canceling a Mandate Canceling a Mandate
PCM_OP_CUST_CHANGE_BUSINESS_PROFILE	Changing a Bill Unit's Business Profile
PCM_OP_CUST_COMMIT_CUSTOMER	Creating Accounts Adding Custom Account Creation Steps Before the Account Is Committed Adding Custom Account Creation Steps After the Account Is Committed Sending Account Information to Your Application when an Account is Created Selecting a Database Schema Customizing Login Names Creating Accounts with Backdated Services or Balances Creating Customer's Collections Profiles at the Time of Customer Account Creation Creating a Service Group Configuring Extended Rating Attributes for a Service Group Associating a Device with a Service Group Assigning Bill Units to Business Profiles Transaction Handling During Account Creation
PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE	Creating Business Profiles
PCM_OP_CUST_CREATE_BAL_GRP	Creating Accounts
PCM_OP_CUST_CREATE_BILLINFO	Creating Accounts Creating Business Profiles
PCM_OP_CUST_CREATE_CUSTOMER	Creating Accounts Creating a Service Group
PCM_OP_CUST_CREATE_PAYINFO	Customizing Customer Payment Information Creating Accounts CVV2/CID Fraud Prevention Functionality Registering a Mandate
PCM_OP_CUST_CREATE_PROFILE	Creating Accounts Managing and Customizing Profiles
PCM_OP_CUST_CREATE_SERVICE	Creating Services Creating Accounts
PCM_OP_CUST_DELETE_ACCT	Deleting Accounts

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_DELETE_PAYINFO	Customizing Customer Payment Information Canceling a Mandate Canceling a Mandate
PCM_OP_CUST_DELETE_PROFILE	Managing and Customizing Profiles
PCM_OP_CUST_FIND	Finding Accounts
PCM_OP_CUST_FIND_PROFILE	Retrieving Account Profile Information
PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO	Getting Information about a Business Profile
PCM_OP_CUST_GET_LIFECYCLE_STATES	Getting Life Cycle States
PCM_OP_CUST_INIT_SERVICE	Creating Services Customizing Automatic Account Creation (AAC) Information
PCM_OP_CUST_MODIFY_CUSTOMER	Modifying an Account Sending Account Information to Your Application when an Account is Modified Creating a Service Group Configuring Extended Rating Attributes for a Service Group Associating a Device with a Service Group Transferring Service Groups between Accounts in Different Schemas
PCM_OP_CUST_MODIFY_PAYINFO	Customizing Customer Payment Information CVV2/CID Fraud Prevention Functionality
PCM_OP_CUST_MODIFY_PROFILE	Managing and Customizing Profiles
PCM_OP_CUST_MODIFY_SERVICE	Modifying Services Changing a Bill Unit's Business Profile
PCM_OP_CUST_POL_COMPARE_PASSWD	Implementing Password Encryption
PCM_OP_CUST_POL_DECRYPT_PASSWD	Implementing Password Encryption
PCM_OP_CUST_POL_ENCRYPT_PASSWD	Implementing Password Encryption Creating Passwords
PCM_OP_CUST_POL_EXPIRATION_PASSWD	Customizing Password Expiration
PCM_OP_CUST_POL_GET_CONFIG	Sending Account Information to Your Application when an Account is Created
PCM_OP_CUST_POL_GET_DB_LIST	Getting a List of Database Schemas Selecting a Database Schema
PCM_OP_CUST_POL_GET_DB_NO	Getting a List of Database Schemas Selecting a Database Schema
PCM_OP_CUST_POL_GET_INTRO_MSG	Customizing the Introductory Message
PCM_OP_CUST_POL_GET_POPLIST	Returning a Point-of-Presence (POP) List
PCM_OP_CUST_POL_POST_COMMIT	Adding Custom Account Creation Steps After the Account Is Committed Creating Accounts Customizing the Introductory Message Sending the Introductory Message

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER	Modifying an Account Sending Account Information to Your Application when an Account is Modified
PCM_OP_CUST_POL_PRE_COMMIT	Adding Custom Account Creation Steps Before the Account Is Committed Creating Accounts
PCM_OP_CUST_POL_PRE_DELETE_PAYINFO	Customizing Customer Payment Information
PCM_OP_CUST_POL_PREP_AACINFO	Customizing Automatic Account Creation (AAC) Information
PCM_OP_CUST_POL_PREP_ACCTINFO	Customizing Account Numbers
PCM_OP_CUST_POL_PREP_INHERITED	Creating Services
PCM_OP_CUST_POL_PREP_LOCALE	Customizing Locale Information
PCM_OP_CUST_POL_PREP_LOGIN	Customizing Login Names Requiring Login Names for Email and Broadband Services Creating Logins for Prepaid Services Creating Logins for Email Services
PCM_OP_CUST_POL_PREP_NAMEINFO	Customizing Name and Address Information
PCM_OP_CUST_POL_PREP_PASSWD	Creating Passwords
PCM_OP_CUST_POL_PREP_PAYINFO	Customizing Payment Method Data Preparation Specifying the Payment Processor Vendor
PCM_OP_CUST_POL_PREP_PROFILE	Managing and Customizing Profiles
PCM_OP_CUST_POL_PREP_STATUS	Customizing Status Changes
PCM_OP_CUST_POL_VALID_AACINFO	Customizing Automatic Account Creation (AAC) Information
PCM_OP_CUST_POL_VALID_ACCTINFO	Customizing Account Numbers
PCM_OP_CUST_POL_VALID_LOCALE	Customizing Locale Information
PCM_OP_CUST_POL_VALID_LOGIN	Customizing Login Names Requiring Login Names for Email and Broadband Services Creating Logins for Prepaid Services
PCM_OP_CUST_POL_VALID_NAMEINFO	Customizing Name and Address Information Creating Custom Country Aliases
PCM_OP_CUST_POL_VALID_PASSWD	Creating Passwords
PCM_OP_CUST_POL_VALID_PAYINFO	Customizing Payment Method Validation Verifying the Maximum Number of CVV2 Digits Registering a Mandate
PCM_OP_CUST_POL_VALID_PROFILE	Managing and Customizing Profiles Configuring Extended Rating Attributes for a Service Group
PCM_OP_CUST_POL_VALID_STATUS	Customizing Status Changes
PCM_OP_CUST_PREP_CUSTOMER	Creating Accounts
PCM_OP_CUST_SET_ACCTINFO	Customizing Account Numbers
PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE	Assigning Bill Units to Business Profiles
PCM_OP_CUST_SET_BAL_GRP	Assigning Bill Units to Business Profiles

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_SET_BILLINFO	Transferring Service Groups between Accounts in the Same Schema Assigning Bill Units to Business Profiles
PCM_OP_CUST_SET_DISCOUNT_STATUS	Backdating Status Changes
PCM_OP_CUST_SET_LOCALE	Customizing Locale Information
PCM_OP_CUST_SET_LOGIN	Customizing Login Names Modifying Services
PCM_OP_CUST_SET_NAMEINFO	Managing Name and Address Information Creating Accounts
PCM_OP_CUST_SET_PASSWD	Creating Passwords Modifying Services
PCM_OP_CUST_SET_PAYINFO	Customizing Customer Payment Information Registering a Mandate Transferring Service Groups between Accounts in the Same Schema
PCM_OP_CUST_SET_PRODUCT_STATUS	Backdating Status Changes
PCM_OP_CUST_SET_STATUS	Changing the Status of an Account, Bill Unit, or Service Customizing Status Changes Inactivating Accounts that Exceed a Specified Limit Backdating Status Changes Creating Accounts Modifying Services Getting Life Cycle States
PCM_OP_CUST_SETUP_TOPUP	Creating Accounts
PCM_OP_CUST_UPDATE_CUSTOMER	Modifying an Account Changing a Bill's Payment Method Customizing Customer Payment Information
PCM_OP_CUST_UPDATE_SERVICES	Modifying Services Changing the Status of an Account, Bill Unit, or Service Backdating Status Changes Getting Life Cycle States Associating a Device with a Service Group Canceling a Service Group Transferring Service Groups between Accounts in Different Schemas
PCM_OP_CUST_VALIDATE_CUSTOMER	Customizing Login Names Customizing Passwords Managing and Customizing Profiles Customizing Payment Method Data Preparation Customizing Payment Method Validation
PCM_OP_DELIVERY_MAIL_SENDMSGS	Sending the Introductory Message

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEVICE_ASSOCIATE	Associating a Device with a Service Group Creating Services Modifying Services
PCM_OP_GROUP_ADD_MEMBER	Managing ACL Groups
PCM_OP_MAIL_DELIV_VERIFY	Customizing Authentication Checks
PCM_OP_MAIL_LOGIN_VERIFY	Customizing Authentication Checks
PCM_OP_PERM_ACL_GROUP_ADD_MEMBER	Managing ACL Groups
PCM_OP_PERM_ACL_GROUP_CREATE	Managing ACL Groups
PCM_OP_PERM_ACL_GROUP_DELETE	Managing ACL Groups
PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER	Managing ACL Groups
PCM_OP_PERM_ACL_GROUP_MODIFY	Managing ACL Groups
PCM_OP_PERM_FIND	Finding CSR Membership
PCM_OP_PUBLISH_GEN_PAYLOAD	Adding Members to a Profile Sharing Group
PCM_OP_PYMT_CHARGE	CVV2/CID Fraud Prevention Functionality
PCM_OP_PYMT_COLLECT	Creating Accounts
PCM_OP_PYMT_GET_ACH_INFO	Customizing Account Creation Charges
PCM_OP_PYMT_POL_SPEC_COLLECT	Customizing Account Creation Charges
PCM_OP_PYMT_POL_SPEC_VALIDATE	Creating Accounts Customizing the Account Used for Credit Card Validation
PCM_OP_PYMT_VALIDATE	Creating Accounts
PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION	Canceling a Service Group
PCM_OP_SUBSCRIPTION_CYCLE_ARREARS	Creating Accounts with Backdated Services or Balances
PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Creating Accounts with Backdated Services or Balances
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP	Deleting Members and Profiles from a Profile Sharing Group Deleting a Profile Sharing Group
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY	Adding Members to a Profile Sharing Group
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS	Adding Members to a Profile Sharing Group
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE	Adding Members to a Profile Sharing Group
PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION	Transferring Service Groups between Accounts in the Same Schema Transferring Service Groups between Accounts in Different Schemas
PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS	Creating a Profile Sharing Group Validating Profile Sharing Group Members
PCM_OP_SUBSCRIPTION_PURCHASE_DEAL	Creating Accounts
PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER	Transferring Service Groups between Accounts in Different Schemas

Table 9-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO	Transferring Service Groups between Accounts in the Same Schema Transferring Service Groups between Accounts in Different Schemas
PCM_OP_SUBSCRIPTION_SET_PRODINFO	Transferring Service Groups between Accounts in the Same Schema Transferring Service Groups between Accounts in Different Schemas
PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE	Creating a Profile Sharing Group Validating Profile Sharing Group Members
PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE	Deleting Members and Profiles from a Profile Sharing Group Deleting a Profile Sharing Group Transferring Service Groups between Accounts in Different Schemas
PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY	Modifying a Profile Sharing Group Validating Profile Sharing Group Members Adding Profiles to a Profile Sharing Group Deleting Members and Profiles from a Profile Sharing Group
PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT	Changing the Owner of a Profile Sharing Group
PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION	Transferring Service Groups between Accounts in the Same Schema Transferring Service Groups between Accounts in Different Schemas
PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY	Creating Accounts
PCM_OP_TCF_PROV_CREATE_SVC_ORDER	Adding Members to a Profile Sharing Group

About Account Locking

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

Transaction Handling During Account Creation

When you use PCM_OP_CUST_COMMIT_CUSTOMER, the value of its PIN_FLD_TXN_FLAGS input flist field determines how it handles transactions:

- When set to **1**, PCM_OP_CUST_COMMIT_CUSTOMER *opens its own transaction* because it authorizes credit cards. Operations related to credit card authorization require that the operation never be rolled back. As soon as the credit card authorization occurs, the transaction is committed before the opcode finishes its operation, making the transaction independent of the operations after the commit.

  This is the default when JCA Resource Adapter is not in XA Transaction mode or when PCM_OP_CUST_COMMIT_CUSTOMER is not called by JCA Resource Adapter.

> **Note:**
>
> After credit card authorization, if errors occur in the account creation processes, an account might be partially created.

- When set to **2**, PCM_OP_CUST_COMMIT_CUSTOMER *does not open its own transaction*. Instead, it supports global transactions opened by a global transaction manager.

  When JCA Resource Adapter is in XA Transaction mode and receives a request to call PCM_OP_CUST_COMMIT_CUSTOMER, one of the following occurs:

  – If the input flist contains the PIN_FLD_TXN_FLAGS field set to **2**, the adapter calls the opcode in the usual manner.

  – If the PIN_FLD_TXN_FLAGS field is missing, the adapter adds the field to the flist and sets it to **2** before calling the opcode.

  – If the PIN_FLD_TXN_FLAGS field is set to any value other than **2**, the adapter rejects the opcode with the PIN_ERR_BAD_VALUE error for the PIN_FLD_TXN_FLAGS field.

> **Note:**
>
> To enable PCM_OP_CUST_COMMIT_CUSTOMER to support transactions opened by a global transaction manager, the PIN_FLD_TXN_FLAGS field must be set to **2**.

PCM_OP_CUST_COMMIT_CUSTOMER returns an error in the following situations:

- A transaction is already open when PCM_OP_CUST_COMMIT_CUSTOMER tries to open a transaction.
- The input flist PIN_FLD_TXN_FLAGS field is set to any value other than **2** when JCA Resource Adapter receives a request to call PCM_OP_CUST_COMMIT_CUSTOMER in XA Transaction mode. For information about the adapter's transaction modes, see *BRM JCA Resource Adapter* guide.

Creating Accounts

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

For backward compatibility, use the PIN_FLD_VERSION input field to support an older version of BRM. See the discussion about supporting an older version of BRM in *BRM Developer's Guide*.

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

> **Note:**
>
> When PCM_OP_CUST_COMMIT_CUSTOMER is called by JCA Resource Adapter in XA Transaction mode, account creation occurs in a transaction started by a global transaction manager.

The account creation process is:

1. PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_CREATE_CUSTOMER.

   No validation is performed by PCM_OP_CUST_CREATE_CUSTOMER prior to attempting the actual creations, so invalid or missing data results in an **ebuf** error to be returned along with an output flist describing the validation problem, if one exists. In general, the PCM_OP_CUST_CREATE_CUSTOMER input flist should be taken from the output of a call to PCM_OP_CUST_PREP_CUSTOMER to ensure the fields have been properly validated.

   If you use rerating, use PCM_OP_CUST_COMMIT_CUSTOMER to create accounts. Do not call PCM_OP_CUST_CREATE_CUSTOMER directly. PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_POL_PRE_COMMIT, and based on that, the **/profile/event_ordering** object is created, which is used for rerating. If you use PCM_OP_CUST_CREATE_CUSTOMER directly to create accounts, the /profile/event_ordering object is not created.

2. PCM_OP_CUST_CREATE_CUSTOMER calls PCM_OP_CUST_PREP_CUSTOMER to validate the account creation information.

3. PCM_OP_CUST_PREP_CUSTOMER follows all the steps for creating an account object but does not commit the object to the database:

   - Opens a transaction.
   - Creates an **/account** object.
   - Sets account credit limits.
   - Purchases a bundle.
   - Creates a **/service** object.

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

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

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

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

   In addition, PCM_OP_CUST_PREP_CUSTOMER calls PCM_OP_PYMT_POL_SPEC_VALIDATE to determine whether a customer's payment information needs to be validated during account creation. If validation is required, PCM_OP_CUST_PREP_CUSTOMER calls PCM_OP_PYMT_VALIDATE to perform the

operation. If validation fails, the results are transformed to be consistent with the results used to describe account creation failure results and are then returned on the output flist. See "Processing Paymentech Address Validation Return Codes".

4. If the data is valid, PCM_OP_CUST_CREATE_CUSTOMER calls an internal opcode to create the **/account** object.

   The internal opcode creates a generic **/account** object. The account is constructed in the following actions, all of which are performed inside the transaction started by PCM_OP_CUST_COMMIT_CUSTOMER or the global transaction manager. To create the account, PCM_OP_CUST_COMMIT_CUSTOMER does the following:

   - Calls PCM_OP_CUST_CREATE_BILLINFO to create one or more bill units (**/billinfo** objects) and propagates the bill unit with billing information.
   - Calls PCM_OP_CUST_CREATE_BAL_GRP to create one or more **/balance_group** objects and link the balance groups to the account and the appropriate bill unit.
   - Calls PCM_OP_CUST_CREATE_PAYINFO to create a **/payinfo** object and link the payment information to the appropriate bill unit.
   - Calls PCM_OP_CUST_CREATE_PROFILE to create a **/profile** object if the input flist contains profile information.
   - Calls PCM_OP_CUST_SET_NAMEINFO to set the contact information.
   - Calls PCM_OP_CUST_SETUP_TOPUP to set up top-up information, if any, for the account.
   - Calls PCM_OP_CUST_SET_STATUS to set the account status to active.

5. PCM_OP_CUST_CREATE_CUSTOMER calls PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY to validate that the bundles passed from the input flist do not violate any bundle dependencies.

6. PCM_OP_CUST_CREATE_CUSTOMER calls PCM_OP_CUST_CREATE_SERVICE to create **/service** objects.

7. PCM_OP_CUST_CREATE_CUSTOMER calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to purchase bundles.

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

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

8. PCM_OP_CUST_COMMIT_CUSTOMER creates the customer's collections profile at the time of customer account creation.
9. Before committing the transaction, PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_POL_PRE_COMMIT.
10. After committing the transaction, PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_POL_POST_COMMIT. See "Adding Custom Account Creation Steps After the Account Is Committed".
11. PCM_OP_CUST_COMMIT_CUSTOMER:
    - Passes credit card in "Adding Custom Account Creation Steps Before the Account Is Committed" formation to PCM_OP_PYMT_VALIDATE and PCM_OP_PYMT_COLLECT, which collect any credit card payments charged at account creation and validate the credit card information returned by the payment processor.
    - If you use multiple schemas, creates the **/uniqueness** object.
    - Calls PCM_OP_CUST_POL_GET_CONFIG to get information used by client applications. See "Sending Account Information to Your Application when an Account is Created".
    - Calls PCM_OP_CUST_SET_BAL_GRP to create balance groups, set credit limits, and set sub-balance consumption rules. See "How BRM Handles Consumption Rules and Credit Limits".

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

Customizing Account Numbers

Use PCM_OP_CUST_POL_PREP_ACCTINFO and PCM_OP_CUST_POL_VALID_ACCTINFO to customize the account number format.

PCM_OP_CUST_POL_VALID_ACCTINFO validates the fields that are required to create or modify an account based on the input from the calling opcode. This opcode is called by PCM_OP_CUST_POL_PREP_ACCTINFO when an account is being created or modified. If the mandatory fields are not passed in, PCM_OP_CUST_POL_VALID_ACCTINFO reports an error.

PCM_OP_CUST_POL_VALID_ACCTINFO is an empty hook. You must modify this opcode only for special situations.

To run these opcodes, PCM_OP_CUST_SET_ACCTINFO initializes an **/account** object with generic fields passed in on the input flist. PCM_OP_CUST_SET_ACCTINFO calls PCM_OP_CUST_POL_PREP_ACCTINFO and PCM_OP_CUST_POL_VALID_ACCTINFO to prepare and validate account information, and returns the account number based on the customizations made in the opcodes.

Customizing the Introductory Message

You can customize how to choose an introductory message and how to send it:

- To choose the message, use PCM_OP_CUST_POL_GET_INTRO_MSG. The type of account and package selected are passed to the operation on the input flist. This enables different introductory messages to be returned based on the values of these fields. The introductory message is returned on the output flist as an uninterpreted buffer of data. This enables the introductory message to include HTML, graphics, and other complex information.

  PCM_OP_CUST_POL_GET_INTRO_MSG is not called by any opcode.
- To send the message, use PCM_OP_CUST_POL_POST_COMMIT.

PCM_OP_CUST_POL_GET_INTRO_MSG reads the message from the file specified in the Connection Manager (CM) **pin.conf** file **intro_dir** entry.

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

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

In each introductory message, a parameter substitution system enables the message to contain values that are specific to this customer, such as the name of the package they chose to purchase. For example, the following enables you to substitute a package name and a promotional code:

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

The package name and promotional code substitution is handled by the following code in PCM_OP_CUST_POL_GET_INTRO_MSG:

```
/*************************************************************
 * Package Name
 *************************************************************/

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

/*************************************************************
 * Promo Code
 *************************************************************/

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

Sending the Introductory Message

PCM_OP_CUST_POL_POST_COMMIT uses the Email DM PCM_OP_DELIVERY_MAIL_SENDMSGS opcode to send a welcoming message. PCM_OP_DELIVERY_MAIL_SENDMSGS in turn calls the Email DM PCM_OP_CREATE_OBJ.

Customizing Account Creation Charges

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

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

PCM_OP_PYMT_POL_SPEC_COLLECT checks all the account balance groups and specifies the amount for each bill unit (**/billinfo** object).

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

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

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

You can disable the default logic for PCM_OP_PYMT_POL_SPEC_COLLECT by changing the value of the **CCcollect** or **DDcollect** business parameters. Change the value from **enabled** to **disabled**.

Creating Multiple Accounts in One Transaction

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

Creating Accounts with Backdated Services or Balances

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

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

- Sets the PIN_FLD_EFFECTIVE_T field in the **/account** object to the backdated date.
- Sets the purchase, usage, or cycle start date to the backdated date unless these dates are explicitly set to a different date.

- Sets the billing day of month (DOM) to the backdated date unless the **ActgDom** business parameter is used.

BRM does not allow backdating the account creation to a date prior to the general ledger (G/L) posting date.

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

When you backdate account creation, you also backdate the charge offer purchase. If the charge offer includes cycle fees, BRM uses PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_ SUBSCRIPTION_CYCLE_ARREARS to calculate cycle forward and cycle arrears fees for the subscription operations.

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

You can create accounts with the services or balances creation date backdated beyond the account or service creation date by running the **pin_bus_params** utility to change the **SubsDis74BackDateValidations** business parameter. For information about this utility, see *BRM Developer's Guide*.

To create accounts with backdated services or balances:

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

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

3. In the file, change **disabled** to **enabled**:

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

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

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.

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

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

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

Customizing Automatic Account Creation (AAC) Information

To validate automatic account creation data, BRM uses PCM_OP_CUST_POL_PREP_AACINFO and PCM_OP_CUST_POL_VALID_AACINFO.

For example, when registering customers, you can offer different packages to different customers by having customers enter account creation numbers. You can also assign account creation numbers to customers to track marketing information. To specify how to validate account creation numbers, edit the PCM_OP_CUST_POL_VALID_AACINFO policy opcode.

PCM_OP_CUST_POL_PREP_AACINFO prepares automatic account creation (AAC) data for validation. This opcode takes the AAC fields for an **/account** and **/service** object during customer account creation and processes them as necessary to prepare for validation. This opcode can be used to prepare ACC info for online account creation. This opcode is an empty hook.

This opcode is called by PCM_OP_CUST_INIT_SERVICE and PCM_OP_CUST_ACCTINFO.

See *BRM Developer's Reference* guide.

Adding Custom Account Creation Steps Before the Account Is Committed

To run additional steps before an account is committed, use PCM_OP_CUST_POL_PRE_COMMIT.

PCM_OP_CUST_POL_PRE_COMMIT is called by PCM_OP_CUST_COMMIT_CUSTOMER just after the **/account** and **/service** objects have been created and initialized but before the transaction containing those operations has been committed. Therefore, PCM_OP_CUST_POL_PRE_COMMIT cannot alter the **/account** and **/service** objects, but it can cancel the account creation process by returning an **ebuf** error. Therefore, you can include tests that the customer must pass before proceeding.

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

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

PCM_OP_CUST_POL_PRE_COMMIT is an empty hook.

Adding Custom Account Creation Steps After the Account Is Committed

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

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

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

No information is returned as output parameters.

Integrating BRM With an External Account Creation Application

See the following topics:

- Sending Account Information to Your Application when an Account is Created
- Sending Account Information to Your Application when an Account is Modified

Sending Account Information to Your Application when an Account is Created

To send account information to your application after the account is created, use PCM_OP_CUST_POL_GET_CONFIG. This opcode is called by PCM_OP_CUST_COMMIT_CUSTOMER after customer account creation has been successfully performed to specify the configuration data that should be returned to the client software.

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

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

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

Sending Account Information to Your Application when an Account is Modified

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

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

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

Returning a Point-of-Presence (POP) List

To get an account creation point-of-presence (POP) list, use PCM_OP_CUST_POL_GET_POPLIST.

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

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

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

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

- PIN_FLD_PHONE
- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP
- PIN_FLD_FLAGS

The PIN_FLD_FLAGS field contains two flags:

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

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

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

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

- PIN_FLD_PHONE
- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP

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

PCM_OP_CUST_POL_GET_POPLIST is not called by any opcode.

Creating Accounts in a Multischema System

In multischema systems, you can configure BRM to balance account loads among database schemas. You can use policy opcodes to customize how to select schemas during account creation. See the following topics:

- Getting a List of Database Schemas
- Selecting a Database Schema

Getting a List of Database Schemas

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

The default implementation returns the cached list of schema distributions created by PCM_OP_CUST_POL_GET_DB_NO during BRM initialization.

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

PCM_OP_CUST_POL_GET_DB_LIST is not called by any opcode.

Selecting a Database Schema

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

You can customize the selection process by customizing PCM_OP_CUST_POL_GET_DB_NO.

Managing Name and Address Information

To add name and address information to a specific account object, use PCM_OP_CUST_SET_NAMEINFO. This opcode sets the fields of the PIN_FLD_NAMEINFO array of a specified **/account** object to the values specified in the input flist PIN_FLD_NAMEINFO array.

PCM_OP_CUST_SET_NAMEINFO does the following:

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

> **Note:**
>
> Element IDs through 100 are reserved for BRM use.

BRM calls PCM_OP_CUST_SET_NAMEINFO as part of the process of creating an **/account** object.

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

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

Customizing Name and Address Information

Use the following opcodes to customize name and address information:

- PCM_OP_CUST_POL_PREP_NAMEINFO
- PCM_OP_CUST_POL_VALID_NAMEINFO

See *BRM Developer's Guide*.

PCM_OP_CUST_SET_NAMEINFO calls PCM_OP_CUST_POL_PREP_NAMEINFO to prepare the customer information for validation and to determine whether the account being created is the BRM payment suspense account. If the country is not provided, it is assumed to be “USA" and “USA" is added as the country value. You can change the **country** parameter in the **pin.conf** file to insert any country when none is provided.

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

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

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

PCM_OP_CUST_SET_NAMEINFO then calls PCM_OP_CUST_POL_VALID_NAMEINFO to validate the information.

Customizing Locale Information

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

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

To customize how locales are set, use the following opcodes:

- To customize the creation of the locale, use PCM_OP_CUST_POL_PREP_LOCALE. This opcode is called by PCM_OP_CUST_SET_LOCALE. This opcode is an empty hook. See *BRM Developer's Guide*.
- To customize how to validate the locale, use PCM_OP_CUST_POL_VALID_LOCALE. See *BRM Developer's Guide*. PCM_OP_CUST_POL_VALID_LOCALE does the following:
  - Validates limit information for a service.
  - Confirms that the value of PIN_FLD_LOCALE in the input flist is one of the valid BRM locales.

Creating Custom Country Aliases

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

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

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

Authenticating and Authorizing Customers

See the following topics:

- Customizing Login Names
- Requiring Login Names for Email and Broadband Services
- Creating Logins for Prepaid Services
- Creating Logins for Email Services
- Creating Passwords
- Authenticating Customers

Customizing Login Names

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

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

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

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

- To prepare login names for validation, use PCM_OP_CUST_POL_PREP_LOGIN. This opcode takes the login field for a service object during customer account creation and processes it as necessary to prepare for validation. For example, you can add characters or change the case to all lowercase. This opcode is called by PCM_OP_CUST_COMMIT_CUSTOMER and PCM_OP_CUST_SET_LOGIN. See *BRM Developer's Guide*.
- To validate login names, use PCM_OP_CUST_POL_VALID_LOGIN (for example, to ensure that they have the required number of characters). This opcode takes the login field for a **/service** object during customer account creation or administrative update and validates it. This opcode is called by PCM_OP_CUST_SET_LOGIN, PCM_OP_CUST_COMMIT_CUSTOMER, and PCM_OP_CUST_VALIDATE_CUSTOMER. See *BRM Developer's Guide*.

> **Caution:**
>
> BRM requires unique login names for service types. BRM will not function properly if PCM_OP_CUST_POL_VALID_LOGIN is customized to allow nonunique login names.

> **Note:**
>
> BRM requires logins for services (they cannot be NULL). BRM requires that login names be unique. BRM will not function properly if PCM_OP_CUST_POL_VALID_LOGIN is customized to allow nonunique login names.

Requiring Login Names for Email and Broadband Services

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

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

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

Creating Logins for Prepaid Services

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

In cases where a prepaid service login is needed, such as Self-Care Manager, the customer's Mobile Station Integrated Services Digital Network (MSISDN) number is stored in the service object PIN_FLD_ALIAS_LIST array.

PCM_OP_CUST_POL_VALID_LOGIN enforces that a password for a service cannot be changed when the account is created or when a service is added. The password can be changed later.

Creating Logins for Email Services

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

Creating Passwords

To add or change passwords, use PCM_OP_CUST_SET_PASSWD.

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

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

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

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

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

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

Customizing Passwords

Use the following opcodes to customize passwords:

- To prepare account or service passwords for validation, use PCM_OP_CUST_POL_PREP_PASSWD. This opcode is called by PCM_OP_CUST_SET_PASSWD. PCM_OP_CUST_POL_PREP_PASSWD takes the password field for an **/account** or **/service** object during customer account creation and prepares it for validation.

  If a service password is passed in, the operation does nothing. If there is no password for a service, it generates one. A password is considered to be passed in if the PIN_FLD_PASSWD_CLEAR field is in the input flist, even if it is NULL.
- To validate account or service passwords, use PCM_OP_CUST_POL_VALID_PASSWD. This opcode is called by PCM_OP_CUST_SET_PASSWD and PCM_OP_CUST_VALIDATE_CUSTOMER. PCM_OP_CUST_POL_VALID_PASSWD takes the password field for an **/account** or **/service** object during customer account creation or administrative update and validates it. The default check is to make sure the password is not NULL and is less than 255 characters.

Implementing Password Encryption

Use the following opcodes to implement password encryption:

PCM_OP_CUST_POL_ENCRYPT_PASSWD encrypts a cleartext password based on the type of the account or service POID given and/or the requested encryption algorithm. The binary result is stored as an ASCII-like string to facilitate storage. All encryption requests from IP accounts get clear text encryption (to support Challenge Handshake Authentication Protocol, or CHAP). Advanced Encryption Standard (AES) or OZT encryption is used for all others. See "About Encrypting Passwords". This opcode is called by PCM_OP_CUST_SET_PASSWD and PCM_OP_CUST_POL_COMPARE_PASSWD.

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

PCM_OP_CUST_POL_DECRYPT_PASSWD decrypts a cleartext password.

Creating Passwords for Prepaid Services

For prepaid services, the password is generated automatically. By default, PCM_OP_CUST_POL_PREP_PASSWD generates a random eight-character password, but you can customize the password algorithm.

Customizing Password Expiration

To customize password expiration, use PCM_OP_CUST_POL_EXPIRATION_PASSWD.

This opcode is called by PCM_OP_CUST_SET_PASSWD.

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

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

To change the default password expiry duration, edit the **passwd_age** entry in the CM **pin.conf** file. For example, instead of 90 days you can set the expiration duration to 150 days.

If successful, PCM_OP_CUST_POL_EXPIRATION_PASSWD returns:

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

Authenticating Customers

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

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

Authenticating User Actions

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

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

1. Calls PCM_OP_ACT_FIND to retrieve the user's **/account** and **/service** objects. See "Finding a Customer's Account Information".
2. Calls PCM_OP_ACT_POL_SPEC_VERIFY to retrieve the list of authentication checks for the specified action. See "Customizing Authentication Checks".
3. Performs all authentication checks specified by the opcode.
4. Returns the following, depending on the success of the transaction:
   - If authentication succeeds, PCM_OP_ACT_FIND_VERIFY returns the POID of the service object and the PIN_FLD_RESULT field set to one of the following:
     - **0** to indicate a valid login and password.
     - **4** to indicate that the customer used a temporary password.
   - If authentication fails, PCM_OP_ACT_FIND_VERIFY returns PIN_FLD_RESULT set to one of the following:
     - **2** to indicate an incorrect password.
     - **5** to indicate that the customer used an expired password.
     - **6** to indicate that the password is no longer valid.

Customizing Authentication Checks

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

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

Table 9-2 lists the available authentication checks:

Table 9-2 Default Authentication Checks

Authentication check	Enum value	Description
PIN_ACT_CHECK_UNDEFINED	0	None
PIN_ACT_CHECK_ACCT_TYPE	1	Check type of **/account** object.
PIN_ACT_CHECK_SRVC_TYPE	4	Check type of **/service** object.
PIN_ACT_CHECK_SRVC_STATUS	5	Check the service status. Can be **active**, **inactive**, or **closed**.
PIN_ACT_CHECK_SRVC_PASSWD	6	Confirm that the password is correct.
PIN_ACT_CHECK_CREDIT_AVAIL	7	Check the available credit.
PIN_ACT_CHECK_DUPE_SESSION	8	Check the open session count for the service. See "Enabling Duplicate Session Checking".

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

The other fields present in an element of the checks array depend on which of the checks is specified by that element.

Table 9-3 shows the authentication checks that PCM_OP_ACT_POL_SPEC_VERIFY returns and their behavior:

Table 9-3 Authentications Checks Returned by PCM_OP_ACT_POL_SPEC_VERIFY

Action	Default authentication checks
PCM_OP_MAIL_DELIV_VERIFY	PIN_ACT_CHECK_SRVC_STATUS = active
PCM_OP_ACT_LOGIN	PIN_ACT_CHECK_SRVC_STATUS = active PIN_ACT_CHECK_SRVC_PASSWD PIN_ACT_CHECK_CREDIT_AVAIL = >=0
PCM_OP_MAIL_LOGIN_VERIFY	PIN_ACT_CHECK_SRVC_STATUS = active PIN_ACT_CHECK_SRVC_PASSWD PIN_ACT_CHECK_CREDIT_AVAIL = >=0
DEFAULT (for everything else)	PIN_ACT_CHECK_SRVC_STATUS = active PIN_ACT_CHECK_SRVC_PASSWD

You can customize PCM_OP_ACT_POL_SPEC_VERIFY to:

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

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

To specify a different set of authentication checks for a new action:

1. Edit the *BRM_home***/source/sys/fm_policy/fm_act_pol/fm_act_pol_spec_vrfy.c** source file.
2. Create a new Activity Policy Facilities Module (FM).

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

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

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

Enabling Duplicate Session Checking

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

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

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

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

Managing and Customizing Profiles

To manage and customize profiles:

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

  This opcode calls other opcodes that prepare and validate the account data. After the account data is validated, PCM_OP_CUST_CREATE_PROFILE calls base opcodes to create a **/profile** object that contains extra information that you want to store about an account and associates it with the account object POID.

  Only one element can be passed in the PIN_FLD_PROFILES array. Otherwise, the opcode ignores the array.

  When automatic rerating is enabled, PCM_OP_CUST_CREATE_PROFILE triggers automatic rerating of backdated extended rating attribute (ERA) modifications when certain conditions are met.

- To modify a profile object, use PCM_OP_CUST_MODIFY_PROFILE. When automatic rerating is enabled, this opcode triggers automatic rerating of backdated ERA modifications when certain conditions are met.

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

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

  Only one element can be passed in the PIN_FLD_PROFILES array. Otherwise, the opcode ignores the array.

- To delete profiles, use PCM_OP_CUST_DELETE_PROFILE.

  If the profile object specified in the input flist is a part of a profile sharing group (**/group/sharing/profile** object), PCM_OP_CUST_DELETE_PROFILE does not delete the profile and returns an error. If the specified profile object is not part of a profile sharing group, this opcode deletes the profile.

To customize profiles, use the following opcodes:

- Use PCM_OP_CUST_POL_PREP_PROFILE to add custom data to the **/profile** object. This opcode is an empty hook. This opcode is called by PCM_OP_CUST_CREATE_PROFILE and PCM_OP_CUST_MODIFY_PROFILE, and returns the flist that comes in.
- Use PCM_OP_CUST_POL_VALID_PROFILE to validate the data in the **/profile** object. This opcode is an empty hook. This opcode is called by PCM_OP_CUST_CREATE_PROFILE, PCM_OP_CUST_VALIDATE_CUSTOMER, and PCM_OP_CUST_MODIFY_PROFILE, and returns the flist that comes in. If the data is not valid, a list of possible problems is returned.

Retrieving Account Profile Information

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

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

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

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

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

Modifying an Account

To modify an account, use PCM_OP_CUST_UPDATE_CUSTOMER and PCM_OP_CUST_MODIFY_CUSTOMER opcodes.

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

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

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

- If the **validate_deal_dependencies** entry is enabled in the CM **pin.conf** file, PCM_OP_CUST_MODIFY_CUSTOMER verifies that any set bundle dependencies are valid. If not, the operation is disallowed.

> **Note:**
>
> The opcode does not validate any objects. You must customize your client application to validate the objects before calling the PCM_OP_CUST_MODIFY_CUSTOMER opcode.

- Purchase the bundle associated with the package.
- Set the account credit limit for each balance of the added package.
- Create service objects associated with the package.
  - If a service with an add-on package is added, and the **ValidateDiscountDependency** business parameter is set to **enabled**, PCM_OP_CUST_MODIFY_CUSTOMER checks for any mutually exclusive relationships between discounts associated with the add-on package and any discounts already owned by the account. If any such relationship is found, the service addition is not continued.

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

  - If the PIN_FLD_STATUS_FLAGS field is set to PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE, PCM_OP_CUST_MODIFY_CUSTOMER verifies that the service group relationships are valid and associates member services with the appropriate balance group.
  - If a subscription service is being added, PCM_OP_CUST_MODIFY_CUSTOMER verifies that it is not a member of another service group.
  - If the *account status* is inactive and the provisioning flag is not set, an error value is returned and PCM_OP_CUST_MODIFY_CUSTOMER exits without modifying the account.
  - If a subscription member service is being created, PCM_OP_CUST_MODIFY_CUSTOMER verifies that the subscription service is not inactive or closed. If the member service requires its own balance group, PCM_OP_CUST_MODIFY_CUSTOMER creates the balance group. Otherwise, it associates the member service with the subscription service's balance group.
- Optionally associate **/device** objects with the services.
- Create notification events to mark the beginning and end of its execution.

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

PCM_OP_CUST_MODIFY_CUSTOMER is an empty hook to PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER, which you can use to export customer data to an external or legacy system for processing.

Table 9-4 lists the opcodes to which PCM_OP_CUST_MODIFY_CUSTOMER passes information from arrays in its input flist. The opcodes, in turn, use the information to create or modify objects:

Table 9-4 Opcodes and Information Received to Create or Modify Objects

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

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

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

Changing a Bill's Payment Method

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

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

  For example:

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

```
1    PIN_FLD_PAYINFO_OBJ POID      [0] 0.0.0.1 /payinfo/cc 4780 0
...
```

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

  For example:

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

  In the preceding example, the new **/payinfo** object is created and associated with the bill unit specified in the input flist.

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

Creating Services

To create a service object BRM uses PCM_OP_CUST_CREATE_SERVICE.

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

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

- Calls PCM_OP_CUST_INIT_SERVICE to create and initialize the **/service** object.

  PCM_OP_CUST_INIT_SERVICE initializes a service in a defunct state with generic fields provided by the input flist. Returns a short flist with the new POID and unencrypted password. This operation is carried out inside a transaction.

  PCM_OP_CUST_INIT_SERVICE calls PCM_OP_CUST_POL_PREP_INHERITED to prepare inherited information to be ready for online account creation. This opcode creates a placeholder for a substruct in a **/service** string when a **/account** and **/service** object is

created. For GSM services, the default BEARER_SERVICE value is an empty string and the default PRIMARY_MSISDN value is set to 0.

- Calls PCM_OP_CUST_SET_PASSWD to encrypt and set a password (if one is supplied) in the **/service** object. If a password is not provided, one is generated. See "Customizing Passwords" for more information.
- Stores any client-supplied inherited information in the **/service** object.
- Calls PCM_OP_CUST_SET_STATUS to set the service status. See "Setting Account, Service, and Bill Unit Status" for more information.
- Calls PCM_OP_DEVICE_ASSOCIATE to associate **/device** objects with the service.

Modifying Services

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

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

PCM_OP_CUST_UPDATE_SERVICES performs these operations:

1. Creates an **/event/notification/service/pre_change** event.
2. If login or alias list information is specified, calls PCM_OP_CUST_SET_LOGIN to update the service login or alias list with the value specified.
3. If password information is specified, calls PCM_OP_CUST_SET_PASSWD to update the service password with the value specified.
4. If service status information is specified, calls PCM_OP_CUST_SET_STATUS to update the status of the service with the value specified.
5. If inherited fields are specified, calls PCM_OP_CUST_MODIFY_SERVICE for most services to update the values for the inherited fields.
6. If one or more PIN_FLD_DEVICES arrays are specified, calls PCM_OP_DEVICE_ASSOCIATE to disassociate or associate the device. Disassociations are processed before associations.
7. Creates an **/event/notification/service/post_change** event.

If successful, the PCM_OP_CUST_UPDATE_SERVICES output flist contains the following:

- The POID of the **/account** object passed in.
- A services array containing the following:
  - The POIDs of the **/service** objects modified.
  - A results array containing the POID of the **/event** objects created to record the changes. The index values returned in the PIN_FLD_RESULTS array correspond to the input flist values as shown in Table 9-5:

Table 9-5 Indexes and Input Flist Values in PIN_FLD_RESULTS

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

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

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

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

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

Finding Accounts

In addition to using the base search opcodes to find accounts, you can use PCM_OP_CUST_FIND. This opcode replaces PCM_OP_SEARCH and PCM_OP_STEP_SEARCH. Given an account number, PCM_OP_CUST_FIND finds the **/account** object in question and returns the POID of the object and any other information that you request. If no results array is specified, the entire contents of the object are returned.

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

Finding a Customer's Account Information

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

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

- If you use a multischema BRM system, it searches all database schemas to find the **/account** object.

  In multischema environments, the calling opcode must not open a transaction before calling PCM_OP_ACT_FIND. Instead, the calling opcode must first call PCM_OP_ACT_FIND to find the schema in which the account resides and then open a transaction on the correct schema.
- If there is an open context (CM connection) to a schema when it is called and it finds the account in a different schema, it switches the context to the correct schema.
- If there is no open context when it is called, it searches all schemas.

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

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

Deleting Accounts

> **Note:**
>
> You can delete accounts in a production system, but ensure that you use this opcode with care.

To delete accounts, use PCM_OP_CUST_DELETE_ACCT. This opcode deletes the specified **/account** object and all related objects (such as events, bill items, bill history, balances, and notes) and disassociates devices that are assigned to the services. It also ensures that all the BRM objects and audit entries containing the subscriber's personal data are purged.

The POID of the object is checked to ensure that the object can be deleted and that the user has permission to delete the object.

This opcode does not delete any audit table entries associated with the **/account** object.

To allow ECE to delete accounts that still have active open session, set the PIN_FLD_FLAG input field to **1.**

You cannot delete the **/account** object if the account was previously associated with a subscription service transfer or if the bill unit it owns is a paying bill unit of a nonpaying bill unit in another account.

> **Note:**
>
> The PCM_OP_CUST_DELETE_ACCT opcode does not delete all custom objects. You can write custom logic to clean up the custom objects in BRM when the **/event/notification/account/pre_delete** and **/event/notification/account/delete** events are generated by the PCM_OP_CUST_DELETE_ACCT opcode.

Customizing Customer Payment Information

To customize how **/payinfo** objects are created, see the following topics:

- Customizing Payment Method Data Preparation
- Customizing Payment Method Validation
- Finding Payment Info
- Customizing the Account Used for Credit Card Validation

Customer payment information is stored in **/payinfo** objects. Use PCM_OP_CUST_SET_PAYINFO to create or modify a **/payinfo** object. This opcode also updates the PIN_FLD_PAYINFO_OBJ field in the **/billinfo** object.

During account modification, PCM_OP_CUST_SET_PAYINFO modifies the payment information for the bill unit if necessary.

PCM_OP_CUST_SET_PAYINFO creates an **/event/audit/customer/payinfo** object.

For credit card payment methods, PCM_OP_CUST_SET_PAYINFO omits the PIN_FLD_SECURITY_ID field from the PCM_OP_CREATE_OBJ input flist when the /payinfo/cc object is created. The result is that the CVV2/CID information is stored in the database with a NULL value.

PCM_OP_CUST_CREATE_PAYINFO is called by PCM_OP_CUST_UPDATE_CUSTOMER when a customer's payment information is modified.

PCM_OP_CUST_SET_PAYINFO is a wrapper opcode that calls the following opcodes:

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

  When calling PCM_OP_CUST_MODIFY_PAYINFO, PCM_OP_CUST_SET_PAYINFO calls PCM_OP_WRITE_FLDS. One or more fields must be selected or an error is returned. The **/payinfo** object is modified only if the data in the input flist is different from the **/payinfo** object data in the database.

  PCM_OP_CUST_MODIFY_PAYINFO omits the PIN_FLD_SECURITY_ID field from the PCM_OP_WRITE_FLDS input flist when the **/payinfo/cc** object is updated. The result is that the CVV2/CID information is stored in the database with a NULL value.

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

PCM_OP_CUST_DELETE_PAYINFO is given the **/payinfo** object POID of the object to delete. You cannot delete a **/payinfo** object that is currently associated with a **/billinfo** object; you must first delete the **/billinfo** object.

Before PCM_OP_CUST_DELETE_PAYINFO deletes the **/payinfo** object, it calls PCM_OP_CUST_POL_PRE_DELETE_PAYINFO to perform any custom actions provided by you (for example, appropriate precautionary actions on a **/payinfo** object marked for deletion if that **/payinfo** object has been customized).

Use PCM_OP_CUST_POL_PRE_DELETE_PAYINFO to perform any actions that may be necessary before a **/payinfo** object is deleted by PCM_OP_CUST_DELETE_PAYINFO. For example, you can provide the code necessary to check if the **/payinfo** object selected for deletion has been customized. And if so, take appropriate precautionary actions before returning that object to the calling PCM_OP_CUST_DELETE_PAYINFO.

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

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

Customizing Payment Method Data Preparation

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

- If the payment method is **invoice**, the **/payinfo/invoice** object is created.
- If the payment method is **cc** (credit card), the **/payinfo/cc** object is created and the information is prepared for online account creation.

  You can specify the valid date format for credit card and debit card expiration.
- If the payment method is **dd** (US and Canadian direct debit), the **/payinfo/dd** object is created and the information is prepared for online account creation.
- If the payment method is **SEPA**, the **/payinfo/sepa** object is created and the information is prepared for SEPA payments.

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

PCM_OP_CUST_POL_PREP_PAYINFO is called by PCM_OP_CUST_VALIDATE_CUSTOMER.

Specifying the Payment Processor Vendor

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

Customizing Payment Method Validation

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

For credit cards, this opcode checks the credit card type, number, expiration date, and CVV2 or CID number during account creation.

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

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

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

PCM_OP_CUST_POL_VALID_PAYINFO is called by PCM_OP_CUST_VALIDATE_CUSTOMER.

Default /payinfo Validation

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

Mandatory fields for validation:

- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP
- PIN_FLD_COUNTRY
- PIN_FLD_CARD_TYPE

Checks performed for credit card-specific billing information:

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

Checks performed for Paymentech direct debit transactions:

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

CVV2/CID Fraud Prevention Functionality

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

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

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

If the **Cvv2Required** business parameter is set to **enabled** (required), these opcodes send the PIN_FLD_SECURITY_ID value directly to the credit card processor when customers add or change credit cards in their accounts. If the CVV2 information is not provided in the input flist, the PIN_FLD_RESULT value is set to PIN_ERR_MISSING_ARG, with the description *"Missing argument"*.

Verifying the Maximum Number of CVV2 Digits

By default, PCM_OP_CUST_POL_VALID_PAYINFO verifies that a credit card's CVV2 passed in the PIN_FLD_SECURITY_ID input flist field does not exceed three digits. You can modify the maximum allowed number of CVV2 digits by customizing the opcode to:

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

Disabling the Credit Card Checksum

To disable the credit card checksum:

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

   ```
   pin_bus_params -r BusParamsCustomer bus_params_customer.xml
   ```

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

   ```
   <CCchecksum>enabled</CCchecksum>
   ```

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

   ```
   pin_bus_params bus_params_customer.xml
   ```

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

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

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

Customizing the Banking Information for US and Canadian Direct Debit

The bank number (**/bank_no**) and debit number (**/debit_num**) fields can be customized to allow for bank and debit numbers that do not meet the US standard (5 digits).

Customizing the Account Used for Credit Card Validation

When validating a credit card at account creation, BRM needs an account to validate the card with. By default, this is the root account. You cannot store this information with each account because the credit card validation is done before the account is created.

Use PCM_OP_PYMT_POL_SPEC_VALIDATE to change the account used for credit card validation.

The default account is:

```
0.0.0.1 /account 1
```

You can change the **1** to some other account number as long as it has a bill type that is not affected by billing operations or events.

PCM_OP_PYMT_POL_SPEC_VALIDATE reads the following CM **pin.conf** file entries:

- The **cc_validate** entry, which specifies whether to validate credit cards.
- The **cc_revalidate** entry, which specifies the amount of time before revalidation is required.

In addition, PCM_OP_PYMT_POL_SPEC_VALIDATE reads the **/config/ach** object to find the merchant value.

The only types of action supported are **prep customer** and **commit customer**. If one of these is not passed in, PIN_ERR_BAD_VALUE is returned. If the action passed in is NULL, a blank flist is returned. If an error occurs, a null flist is returned.

If the PIN_FLD_PAY_TYPE value is NULL, only the POID from the **pin.conf** file is returned. If the PIN_FLD_PAY_TYPE value is not PIN_PAY_TYPE_CC or PIN_PAY_TYPE_DD, the result is set to PIN_BOOLEAN_FALSE because there is nothing to validate. If the value is PIN_PAY_TYPE_CC or PIN_PAY_TYPE_DD, the result is set to PIN_BOOLEAN_TRUE.

Managing Mandate Information

The following sections describe how BRM manages mandate information when you use SEPA processing.

Registering a Mandate

To register the mandate information provided during customer account creation, BRM uses PCM_OP_CUST_SET_PAYINFO.

PCM_OP_CUST_SET_PAYINFO takes as input the mandate information such as the customer's IBAN and BIC and the creditor identification code.

This opcode adds the mandate by:

1. Creating a unique mandate reference number (UMR) for the mandate, if it is not provided in the input flist
2. Setting the status of the mandate to active

3. Creating the **/payinfo/sepa** object with the mandate information

PCM_OP_CUST_SET_PAYINFO calls PCM_OP_CUST_CREATE_PAYINFO, which calls PCM_OP_CUST_POL_VALID_PAYINFO, which validates that the format of the IBAN and BIC comply with the formats described in the SEPA rulebooks. Additional custom validations can be performed on the mandate information, such as country-specific validations.

Updating a Mandate

To update the mandate information, BRM uses the following opcodes:

- PCM_OP_CUST_AMEND_MANDATE, to update the customer information
- PCM_OP_CUST_AMEND_CREDITOR_INFO, to update the creditor information

PCM_OP_CUST_AMEND_MANDATE takes as input a reference to the **/payinfo/sepa** object and the mandate information to update.

This opcode updates the mandate by:

1. Updating the **/payinfo/sepa** object with the new mandate information
2. Updating the PIN_FLD_MANDATE_AMENDED field in the **/payinfo/sepa** object by using flags to indicate the fields that are amended
3. Generating the **/event/activity/sepa/mandate_amendment** event to record the mandate update

PCM_OP_CUST_AMEND_CREDITOR_INFO takes as input the creditor ID to be updated and the new creditor ID and creditor name.

This opcode updates the creditor information by:

1. Updating the **/config/creditor** object with the new creditor information
2. Determining if any **/payinfo/sepa** object exists that is associated with the original creditor ID and updating the **/payinfo/sepa** object with the new creditor information
3. In each **/payinfo/sepa** object that is updated, updating the PIN_FLD_MANDATE_AMENDED field by using flags to indicate the fields that are amended
4. Generating the **/event/activity/sepa/mandate_amendment** event to record the update

Canceling a Mandate

To cancel a mandate, BRM uses PCM_OP_CUST_CANCEL_MANDATE.

PCM_OP_CUST_CANCEL_MANDATE takes as input the reference to the **/payinfo/sepa** object that contains the mandate to cancel.

This opcode cancels the mandate by:

1. Setting the PIN_FLD_MANDATE_STATUS field in the **/payinfo/sepa** object to PIN_MANDATE_STATUS_CANCELED
2. Setting the PIN_FLD_MANDATE_END_T field to the current time to record the time of cancellation
3. Calling PCM_OP_CUST_DELETE_PAYINFO to delete the **/payinfo/sepa** object. The opcode does not cancel the **/payinfo/sepa** object if it determines that it is associated with a bill unit or if a SEPA payment request is pending

Setting Account, Service, and Bill Unit Status

See the following topics:

- Changing the Status of an Account, Bill Unit, or Service
- Customizing Status Changes
- Inactivating Accounts that Exceed a Specified Limit

Changing the Status of an Account, Bill Unit, or Service

PCM_OP_CUST_SET_STATUS changes and checks the status of an account, bill unit, or service. You can call the opcode directly for accounts and bill units. For service status changes, use PCM_OP_CUST_UPDATE_SERVICES, which in turn calls PCM_OP_CUST_SET_STATUS.

> **Note:**
>
> Do not callPCM_OP_CUST_SET_STATUS directly for service status changes. Use PCM_OP_CUST_UPDATE_SERVICES instead.

In addition to changing status, PCM_OP_CUST_SET_STATUS does the following:

- Triggers auto-billing if bills are still pending.
- Calls opcodes to prorate cycle fees.

To return all the fields in the event object, set the PCM_OPFLG_READ_RESULT flag. If this flag is not set, PCM_OP_CUST_SET_STATUS returns only the POID.

If PCM_OP_CUST_SET_STATUS is called at the account level, more than one result can be returned:

- One for each service owned by this account.
- One for each bill unit owned by this account.
- One or more for each child account and nonpaying bill unit.
- One for each deferred action.

To run PCM_OP_CUST_SET_STATUS without changing data in the database, use the PCM_OPFLG_CALC_ONLY flag.

- If the flag is set, no fields in the database are changed and the event object is not created. The fields used to create the event object and adjustment item are returned to the caller.
- If the flag is *not* set, the **/event/customer/status** object is created to record details of the operation.

PIN_FLD_STATUS Field Values for PCM_OP_CUST_SET_STATUS

When you use PCM_OP_CUST_SET_STATUS, set the account or bill unit status entry to one of the following field values listed in Table 9-6.

Table 9-6 PIN_FLD_STATUS Values

PIN_FLD_STATUS field	Description	Value
PIN_STATUS_ACTIVE	Account or bill unit fully functional; login is subject to credit limit check.	**10100**
PIN_STATUS_INACTIVE	Inactive account or bill unit; no logins permitted and no fees are charged.	**10102**
PIN_STATUS_CLOSED	Closes the account or bill unit, which can be reopened later. This does not activate the corresponding charge offers.	**10103**
PIN_STATUS_DEFUNCT	Reserved for BRM. PIN_ERR_BAD_ARG is returned if you try to set a status to this value.	**0**

PIN_FLD_STATUS_FLAG Field Values for PCM_OP_CUST_SET_STATUS

The values used to further define the status for the account, bill unit, or service are listed in Table 9-7:

Table 9-7 PIN_FLD_STATUS_FLAG Values

PIN_FLD_STATUS_FLAG field	Description	Value
PIN_STATUS_FLAG_ACTIVATE	Sets the future activation date (accounts and bill units only).	**0x01**
PIN_STATUS_FLAG_DEBT	Displays the outstanding debt (accounts and bill units only).	**0x02**
PIN_STATUS_FLAG_MANUAL	Displays manual operations performed by the administrative operator.	**0x04**
PIN_STATUS_FLAG_DUE_TO_ACCOUNT	Closes all related services for the account or bill unit (accounts and bill units only).	**0x08**
PIN_STATUS_FLAG_DUE_TO_PARENT	Closes all child accounts if the parent account is closed (only for accounts in groups set up for billing purposes). Closes all child bill units if the parent bill unit is closed.	**0x10**
PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE	Changes all member services when the subscription's service status is changed.	**0x20000**
PIN_STATUS_FLAG_PO_EXHAUSTED	Obsolete.	**0x20**
PIN_STATUS_FLAG_PROVISIONING	Identifies tasks that use the provisioning system.	**0x40**

The field values in the object are set as follows:

- If the input status is PIN_STATUS_ACTIVE:
  - Flags = (Old flag AND NOT (New flag))
  - If the result flag is **0**, new status is input status, else old status.
- If the input status is PIN_STATUS_INACTIVE:
  - The **status** field is set to inactive.

  - The **flags** field is set to the **OR** of the old flags in the database and the new **flags** on the input flist.
- If the input status is PIN_STATUS_CLOSED:
  - The **status** field is set to **closed**.
  - The **flags** field is set to the **OR** of the old flags and new flags.

If the status change is caused by a change to the parent account or bill unit, PIN_FLD_STATUS_FLAG_DUE_TO_PARENT is set in the new flags.

If the status change is caused by a change to an accounts receivable (AR) account or AR bill unit, PIN_FLD_STATUS_FLAG_DUE_TO_ACCOUNT is set in the new flags.

Accounts, bill units, or services marked as **closed** (terminated) are actually closed when the **pin_deferred_act** billing application is run.

About Deferred Actions When Using PCM_OP_CUST_SET_STATUS

For deferred actions, the PIN_FLD_WHEN_T field is passed to indicate the date on which the status change is made. It also creates a **/schedule** object to store the input flist to call PCM_OP_CUST_SET_STATUS on the specified date.

You can create, delete, run, and modify **/schedule** objects by calling the opcodes in Table 9-8:

Table 9-8 Opcodes used to Modify /schedule Objects

Opcode	Function
PCM_OP_ACT_SCHEDULE_CREATE	Creates a schedule object.
PCM_OP_ACT_SCHEDULE_DELETE	Deletes a schedule object.
PCM_OP_ACT_SCHEDULE_MODIFY	Modifies a schedule object.
PCM_OP_ACT_SCHEDULE_EXECUTE	Runs a schedule object. This opcode is called directly by the **pin_deferred_act** billing utility.

By default, the **pin_deferred_act** utility, which is part of the **pin_bill_day** billing script, finds expired schedule objects and calls PCM_OP_ACT_SCHEDULE_EXECUTE, which in turn calls PCM_OP_CUST_SET_STATUS.

If the deferred status change is for closing an account, bill unit, or service, PCM_OP_CUST_SET_STATUS sets the PIN_FLD_CLOSE_WHEN_T field in the account, bill unit, or service to midnight on the specified date.

Setting, Resetting, or Unsetting a Deferred CLOSE

When setting or resetting a deferred CLOSE, PCM_OP_CUST_SET_STATUS calls a standard opcode to set the PIN_FLD_PURCHASE_END_T, PIN_FLD_CYCLE_END_T, and PIN_FLD_USAGE_END_T fields for each corresponding charge offer and discount offer to the deferred CLOSE date if the old values for these fields are later than the deferred CLOSE date or if these fields are set to NEVER (“0").

If a deferred CLOSE is canceled (the schedule object is deleted), PCM_OP_CUST_SET_STATUS calls a standard opcode to set the values of the PIN_FLD_PURCHASE_END_T, PIN_FLD_CYCLE_END_T, and PIN_FLD_USAGE_END_T fields for all corresponding charge offers and discount offers to the old values from the previous **/event/billing/product/action/modify** event.

Customizing Status Changes

You can customize status changes by using the following opcodes:

- To customize the way status is changed for an account or service, use PCM_OP_CUST_POL_PREP_STATUS. This call is used to modify status information before changing the account. For example, you can use this opcode to customize how customer service representatives (CSR) can set permissions on specific service types. This opcode is called by PCM_OP_CUST_SET_STATUS. This opcode is an empty hook.
- To validate status changes for an account or service, use PCM_OP_CUST_POL_VALID_STATUS. This opcode is called by PCM_OP_CUST_SET_STATUS. The default is to do no additional checking and to return the verified information.

See *BRM Developer's Guide*.

Inactivating Accounts that Exceed a Specified Limit

Use PCM_OP_ACT_POL_EVENT_LIMIT to inactivate any account or account hierarchy that exceeds a specified limit.

PCM_OP_ACT_POL_EVENT_LIMIT performs the following tasks:

1. Calls PCM_OP_CUST_SET_STATUS to inactivate an account or account hierarchy based on the status set in the PIN_FLD_STATUS field.
2. Calls PCM_OP_ACT_POL_EVENT_NOTIFY to notify any custom applications that a limit was reached and an action took place.

PCM_OP_ACT_POL_EVENT_LIMIT is not called by any opcode.

Backdating Status Changes

To backdate a status change, enter the date to which you want to backdate the account, service, charge offer, or discount offer status change in the PIN_FLD_END_T field of the applicable opcodes. For example, enter the backdated date in the PIN_FLD_ END_T field of the following opcodes:

- PCM_OP_CUST_SET_STATUS: for backdating the account status change.
- PCM_OP_CUST_UPDATE_SERVICES: for backdating the service status change.
- PCM_OP_CUST_SET_PRODUCT_STATUS: for backdating the charge offer status change.
- PCM_OP_CUST_SET_DISCOUNT_STATUS: for backdating the discount status change.

Getting Life Cycle States

PCM_OP_CUST_GET_LIFECYCLE_STATES gets one of the following, depending on what information is passed to it:

- If only the account and service POIDs are passed, gets the life cycle states configuration object (**/config/lifecycle_states**) associated with a specified service type
- If the account and service POIDs and the state ID are passed, gets the PIN_FLD_STATES array for that state from a life cycle states configuration object

- If the account, service, and bill unit (**/billinfo** object) POIDs are passed, gets the PIN_FLD_STATES array for the initial state (PIN_FLD_INITIAL_STATE = 1) of the life cycle

PCM_OP_CUST_GET_LIFECYCLE_STATES is called by the following components:

- **pin_state_change** calls this opcode to get state transition information from a life cycle states configuration object.
- PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE calls this opcode to get the state expiration time. After a balance is adjusted for a service that uses a custom life cycle, PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE triggers any required service state change and updates the state expiration date in the **/service** object.
- PCM_OP_CUST_SET_STATUS calls this opcode to get the service's initial state.
- PCM_OP_CUST_SET_STATUS and PCM_OP_CUST_UPDATE_SERVICES opcodes call this opcode during validation.

PCM_OP_CUST_GET_LIFECYCLE_STATES is used only if the **SubscriberLifeCycle** business parameter is associated with the specified service's bill unit and is enabled.

Amending Creditor Information

To amend creditor information, use PCM_OP_CUST_AMEND_CREDITOR_INFO.

PCM_OP_CUST_AMEND_CREDITOR_INFO takes as input the creditor ID to be updated and the new creditor ID and creditor name.

PCM_OP_CUST_AMEND_CREDITOR_INFO does the following:

1. Updates the **/config/creditor** object with the new creditor information.
2. Determines if any **/payinfo/sepa** object exists that is associated with the original creditor ID and updating the **/payinfo/sepa** object with the new creditor information.
3. In each **/payinfo/sepa** object that is updated, updates the PIN_FLD_MANDATE_AMENDED field by using flags to indicate the fields that are amended.
4. Generates the **/event/activity/sepa/mandate_amendment** event to record the update.

Amending a Mandate

To amend a mandate, BRM calls PCM_OP_CUST_AMEND_MANDATE.

PCM_OP_CUST_AMEND_MANDATE takes as input a reference to the **/payinfo/sepa** object and the mandate information to update.

PCM_OP_CUST_AMEND_MANDATE does the following:

1. Updates the **/payinfo/sepa** object with the new mandate information.
2. Updates the PIN_FLD_MANDATE_AMENDED field in the **/payinfo/sepa** object by using flags to indicate the fields that are amended.
3. Generates the **/event/activity/sepa/mandate_amendment** event to record the mandate update.

Canceling a Mandate

To cancel a mandate, use PCM_OP_CUST_CANCEL_MANDATE.

PCM_OP_CUST_CANCEL_MANDATE takes as input the reference to the **/payinfo/sepa** object that contains the mandate to cancel.

PCM_OP_CUST_CANCEL_MANDATE does the following:

1. Sets the PIN_FLD_MANDATE_STATUS field in the **/payinfo/sepa** object to PIN_MANDATE_STATUS_CANCELED.
2. Sets the PIN_FLD_MANDATE_END_T field to the current time to record the time of cancellation.
3. Calls PCM_OP_CUST_DELETE_PAYINFO to delete the **/payinfo/sepa** object. The opcode does not cancel the **/payinfo/sepa** object if it determines that it is associated with a bill unit or if a SEPA payment request is pending.

Managing Deferred Actions

See the following topics:

- Scheduling Deferred Actions
- Modifying Deferred Actions
- Deleting Deferred Actions
- Executing Deferred Actions
- Performing Policy Checks before Scheduling Deferred Actions

Scheduling Deferred Actions

PCM_OP_ACT_SCHEDULE_CREATE creates **/schedule** objects, which schedule a single action for a predetermined date and time.

BRM supports the following deferred actions:

- Account activation, deactivation, and closure.
- Account parent change.
- Service activation and closure.

Two optional fields, PIN_FLD_WHEN_T and PIN_FLD_DESCR, enable deferred actions in BRM. These fields are set by the input flists of PCM_OP_ACT_POL_EVENT_NOTIFY or PCM_OP_BILL_GROUP_MOVE_MEMBER.

- PIN_FLD_DESCR describes the deferred action.
- PIN_FLD_WHEN_T specifies when the deferred action occurs.

If the PIN_FLD_WHEN_T field is populated in the input flist of the calling opcode, that opcode can call PCM_OP_ACT_SCHEDULE_CREATE to create a **/schedule** object to hold the input flist information. The **/schedule** object remains active until the PIN_FLD_WHEN_T time expires and PCM_OP_ACT_SCHEDULE_EXECUTE runs the action stored in the **/schedule** object.

> **Note:**
>
> Deferred actions are always rounded off to midnight. For example, if you use PCM_OP_ACT_SCHEDULE_CREATE to schedule a deferred action on January 15 at 8:00 a.m., the schedule object is created with a scheduled time of January 15 at 00:00.

Modifying Deferred Actions

PCM_OP_ACT_SCHEDULE_MODIFY modifies an existing **/schedule** object.

This opcode modifies the text description in PIN_FLD_DESCR field and the scheduled date in PIN_FLD_WHEN_T field.

Deleting Deferred Actions

PCM_OP_ACT_SCHEDULE_DELETE deletes existing **/schedule** objects.

Executing Deferred Actions

PCM_OP_ACT_SCHEDULE_EXECUTE runs any deferred actions defined in a **/schedule** object. This opcode is called directly by the **pin_deferred_act** billing utility.

When you run the **pin_deferred_act** utility, it searches for any **/schedule** object with both an expired PIN_FLD_WHEN_T field and a PIN_FLD_STATUS field marked **Pending**. When it finds one, it calls PCM_OP_ACT_SCHEDULE_EXECUTE to:

1. Run the action defined in the **/schedule** object.
2. Update PIN_FLD_STATUS to either **Done** or **Error**, depending on its success.

Performing Policy Checks before Scheduling Deferred Actions

Use PCM_OP_ACT_POL_VALIDATE_SCHEDULE to perform custom policy checks before creating a **/schedule** object. For example, you can customize this opcode to verify that an account balance is zero before scheduling it for closure.

This opcode is an empty hook.

This opcode is called by PCM_OP_ACT_SCHEDULE_CREATE.

Managing Service Groups

See the following topics:

- Creating a Service Group
- Configuring Extended Rating Attributes for a Service Group
- Associating a Device with a Service Group
- Adding a Service to an Existing Service Group
- Canceling a Service Group
- Transferring Service Groups between Accounts in the Same Schema

- Transferring Service Groups between Accounts in Different Schemas

Creating a Service Group

A service group is composed of a subscription service and any number of member services. The **/service** objects for subscription services are created when packages are purchased. This can occur at the following times:

- When registering customers, use PCM_OP_CUST_COMMIT_CUSTOMER.
- When customers purchase a package for an existing account, use PCM_OP_CUST_MODIFY_CUSTOMER.

These opcodes call other opcodes to set up a customer's services and create the service objects.

These opcodes set the service group relationship fields in the service objects that they create. Services are created in the following order: account services, the subscription service, and member services in the service group.

To create a service group, set the following fields in the PCM_OP_CUST_CREATE_CUSTOMER and PCM_OP_CUST_MODIFY_CUSTOMER input flists:

- Add a PIN_FLD_SERVICES array for each service in the service group.

> **Note:**
>
> The service arrays must start with array element **1**.

- In each service array, set the PIN_FLD_SUBSCRIPTION_OBJ field to specify the POID of the subscription service. If the service being created *is* the subscription service, this field and the PIN_FLD_POID field specify the same value.
- Add a PIN_FLD_BAL_INFO array to create a balance group for the subscription service. Optionally, specify a credit limit or threshold.
- Add an additional PIN_FLD_BAL_INFO array for each member service that should have its own balance group, and optionally set the credit limit and threshold. A member service that does not have a balance group uses the subscription service's balance group.
- In each PIN_FLD_SERVICES array, set the PIN_FLD_BAL_INFO_INDEX field to specify the index of the array that contains the balance group for that service.

Configuring Extended Rating Attributes for a Service Group

After setting up systemwide extended rating attribute (ERAs), you assign an ERA to a customer's service and specify customer-specific information (for example, the customer's birthday for a birthday discount ERA).

Service ERAs are stored in **/profile/serv_extrating** objects. These objects are typically associated with customer accounts at two times:

- When registering customers, use PCM_OP_CUST_COMMIT_CUSTOMER.
- When customers purchase a package for an existing account, use PCM_OP_CUST_MODIFY_CUSTOMER.

These opcodes call other opcodes to create or modify the **/profile** objects.

To configure a service group ERA for a customer, add a PIN_FLD_PROFILES array to the subscription service's PIN_FLD_SERVICES array on the input flist of the *_CUSTOMER opcodes. In the profiles array, set the POID of the **/profile/serv_extrating** object owned by the service.

To configure the ERA with customer-specific data, write custom code that sets the customer's information in the PIN_FLD_DATA array of the **/profile/serv_extrating** object.

To validate the customer's profile information, you must modify PCM_OP_CUST_POL_VALID_PROFILE.

You can also use the customer profile opcodes to create and modify **/profile** objects directly.

Associating a Device with a Service Group

Customers use devices, such as a wireless handset, to access the services in their service groups. The device information, such as device ID, manufacturer, and associated account and service, are stored in a **/device** object. You associate **/device** objects with subscription **/service** objects when you register customers or add packages to customer accounts:

- When registering customers, use PCM_OP_CUST_COMMIT_CUSTOMER.
- When customers purchase a package for an existing account, use PCM_OP_CUST_MODIFY_CUSTOMER.

These opcodes call other opcodes to create or modify the **/device** objects.

To create a **/device** object and associate it with a service group, set the following fields in the subscription service's PIN_FLD_SERVICES array on the input flist of the customer opcodes:

- In the PIN_FLD_DEVICE_OBJ field of the PIN_FLD_DEVICES array, specify the POID of the **/device** object.
- In the PIN_FLD_ALIAS_LIST array, specify the number associated with the device (such as the IMSI or MSISDN).

You can also use PCM_OP_CUST_UPDATE_SERVICES to modify **/device** objects. This opcode calls PCM_OP_DEVICE_ASSOCIATE to associate or disassociate a device with a service.

To create or modify device objects directly, use the PCM_OP_DEVICE* opcodes.

Adding a Service to an Existing Service Group

You add a service to a service group in the same way that you create a service in a service group. Use PCM_OP_CUST_MODIFY_CUSTOMER and set the service information in the input flist. See "Creating a Service Group".

Canceling a Service Group

Use PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION to cancel the services in a service group. This opcode cancels the group's subscription service and all its member services.

Specify the following information in PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION input flist:

- Set PIN_FLD_POID to the POID of the account object that owns the service group.

- Set PIN_FLD_SUBSCRIPTION_OBJ to the POID of the subscription service to be canceled.
- Set the input flag PIN_FLD_FLAGS to PIN_BILL_FLG_BILL_NOW, to bill the account immediately upon service cancellation.

> **Note:**
>
> If the input flag is not set, the account is billed for the service on the regularly scheduled billing date.

PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION performs the following tasks:

1. If the PIN_BILL_FLG_BILL_NOW flag is passed in the input flist, calls PCM_OP_BILL_MAKE_BILL_NOW for each bill unit associated with the service group.

> **Note:**
>
> Typically, all member services are associated with the subscription service's bill unit. However, it is possible for member services to be associated with different bill units.

2. Calls PCM_OP_CUST_UPDATE_SERVICES to update the status of the subscription service and member services. PCM_OP_CUST_UPDATE_SERVICES does the following:
   a. Changes the status of the subscription service and all its member services to *closed*.
   b. Changes the status of all associated with the subscription service and member services to *canceled*.
   c. Deletes any charge or discount sharing groups owned by the subscription service and the member services, and updates the ordered balance group (**/ordered_balgrp** object) of those services.
   d. Removes the subscription service and member services from any charge or discount sharing group of which they are members.
3. Generates the **/event/audit/subscription/cancel** audit event to record the service cancellation.

If the PIN_BILL_FLG_BILL_NOW flag is specified in the input flist, the service is billed upon cancellation. Otherwise, the service is billed during the regularly scheduled billing time.

If successful, PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION returns the following:

- The POID of the subscription service that was canceled.
- The POID of the **/event/audit/subscription/cancel** audit event created to record the cancellation.

Transferring Service Groups between Accounts in the Same Schema

To transfer a service group to another account in the same schema, use PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION.

Specify the following information in the PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION input flist:

- The POID of the source account.
- The POID of the target account.
- The POID of the subscription service to transfer.
- The POID of a bill unit (**/billinfo** object) and of a pay info object (**/payinfo**):
  - If the subscription service is to be associated with an existing bill unit in the target account, specify the POIDs of that bill unit and pay info object.
  - If the subscription service is to be associated with a new bill unit created for the service only, specify the POIDs of the new bill unit and pay info object.
- The POID of an existing pay info object for the target account or the POID of a new pay info object created for the service only.

To transfer the service group's member services, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Sets the account POID to the target account's POID.
2. Creates a transfer list array element, and adds it to the service objects.
3. Creates an **/event/audit/subscription/transfer** audit event to record the service transfer.

To transfer the service group's devices, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Finds all the devices owned by the service, and sets each device's account POID to the target account POID.
2. Creates an **/event/audit/subscription/transfer** audit event to record the device transfer.

To transfer the service group's charge and discount offers, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION cancels the service group offers owned by the source account and creates copies for the target account. To do this, the opcode does the following for each member service:

1. Finds all charge and discount offers associated with the service in the source account.
2. For each charge offer instance:
   a. Calls PCM_OP_SUBSCRIPTION_SET_PRODINFO to set the purchase, cycle, and usage end dates to the transfer date in the source account and to apply cycle forward fees.
   b. Creates another instance of the charge offer and assigns it to the target account with purchase, cycle, and usage start dates set to the transfer date.
   c. Creates an **/event/billing/product/action/purchase** event to record the transfer of the charge offer instance.
3. For each discount offer instance:
   a. Calls PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO to set the purchase, cycle, and usage end dates to the transfer date in the source account and to apply cycle forward fees.
   b. Creates another instance of the discount offer and assigns it to the target account with purchase, cycle, and usage start dates set to the transfer date.
   c. Creates an **/event/billing/discount/action/purchase** event to record the transfer of the discount offer instance.

To transfer the service group's billing information, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION associates each service balance group with an existing bill unit in the target account or with a new bill unit created for the service group. To do this, the opcode does the following:

1. If a **/billinfo** object is specified in the input flist, the opcode sets the PIN_FLD_BILLINFO_OBJ field in the service **/balance_group** object to reference that **/billinfo** object.
2. If a **/payinfo** object is specified in the input flist, the opcode sets the PIN_FLD_PAYINFO_OBJ field in the service **/balance_group** object to reference that **/payinfo** object.
3. If a **/billinfo** object is not specified, the opcode calls PCM_OP_CUST_SET_BILLINFO to create a **/billinfo** object and sets the PIN_FLD_BILLINFO_OBJ field in the service **/balance_group** object to the new **/billinfo** object.
4. If a **/payinfo** object is not specified, the opcode calls PCM_OP_CUST_SET_PAYINFO to create a **/payinfo** object and sets the PIN_FLD_PAYINFO_OBJ field in the service **/balance_group** object to the new **/payinfo** object.

To transfer the service group's sub-balances, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following:

1. Modifies each service **/balance_group** object to reference the **/billinfo** object in the account.
2. Sets the sub-balance validity based on the **SubBalValidity** business parameter.
3. Creates an **/event/audit/subscription/transfer** audit event to record the balance group transfer.

To transfer pending scheduled actions for a service in a service group when transferring the group to another account, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION calls PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION.

The **TransferScheduledActions** business parameter must be enabled for PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION to call PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION. See "Enabling Transfer of Pending Scheduled Actions during Service Group Transfers".

When a service group is transferred to another account, its subscription service and member services are deleted from any sharing group of which they are a member. PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Finds the charge or discount sharing groups that the service is a member of and deletes the service from the sharing group.
2. Deletes any ordered balance group associated with the service.

When a subscription service is transferred to another account, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION finds and deletes any charge or discount sharing group that it owns.

Enabling Transfer of Pending Scheduled Actions during Service Group Transfers

To enable the transfer of pending scheduled actions during service group transfers:

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

```
pin_bus_params -r BusParamsSubscription bus_params_subscription.xm
```

3. In the file, change **disabled** to **enabled**:

```
<TransferScheduledActions>enabled</TransferScheduledActions>
```

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

```
pin_bus_params bus_params_subscription.xml
```

6. Stop and restart the CM.

Transferring Service Groups between Accounts in Different Schemas

To transfer a service group between accounts stored in multiple schemas, you enable the **RecreateDuringSubscriptionTransfer** business parameter.

> **Note:**
>
> When **RecreateDuringSubscriptionTransfer** is enabled, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION re-creates all the objects in the schema to which the service groups are transferred.

Enabling Transfer of Service Groups between Accounts in Multiple Schemas

To enable the transfer of service groups to an account in a different schema:

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

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

3. In the file, change **disabled** to **enabled**:

```
<RecreateDuringSubscriptionTransfer>disabled</RecreateDuringSubscriptionTransfer>
```

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

```
pin_bus_params bus_params_subscription.xml
```

6. Stop and restart the CM.

Transferring Service Groups across Schemas

To transfer a service group, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following:

1. Finds all the related services. If the service to be transferred is a subscription service, this opcode finds all the member services and groups.
2. Changes the status of the services to CANCELLED and recalculates charges for all service-level that use PCM_OP_SUBSCRIPTION_SET_PRODINFO or PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO.
3. Creates an flist of all the services to be transferred, and saves login information, profiles, and devices.

4. Adds login information to the source services by prepending the login details with the service POID, and generates an **/event/customer/login** event.
5. Calls PCM_OP_CUST_MODIFY_CUSTOMER to create new service instances for the account in the target schema.
6. Creates **/billinfo**, **/payinfo**, and **/balance_group** objects if the source objects do not already have them.

> **Note:**
>
> The **/billinfo** object passed in the input flist must belong to the account in the target schema.

7. Fetches the encrypted password from the service in the source schema, sets it in the service in the target schema, and generates an **/event/customer/status** and an **/event/customer/login** event.
8. If called from PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER, saves the **/service**, **/balance_group**, and **/billinfo** objects in the NAMED_TRANS_FLIST array.
9. Updates the SUBSCRIPTION_OBJ field of the new instance of each member service in the target schema with the POID of the subscription service created in the target schema.
10. Updates **/device** object with the service and the account in the target schema.

> **Note:**
>
> Devices associated with the services to be transferred must not be associated with any services that are not transferred.

11. Re-creates the **/profile** objects in the target schema using PCM_OP_CREATE_OBJ that has **effective_t** set to the transfer time.

> **Note:**
>
> To cancel the source service instances, the opcode deletes the **/ordered_balgrp** and **/group/sharing/*** relationships and saves the audit copies on the source service instances for processing late CDRs. Sharing group relationships are terminated on the transfer date, and the new service instances do not participate in any sharing relationships.

12. Calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE to delete all sharing groups in which service is an owner or a member.
13. Creates the **/purchased_discount** and **/purchased_product** objects and cycle forward events. For cycle arrears events, calls PRO_FORCED in CYCLE_FEE_FLAGS.
14. Regenerates a package ID for the offerings created in the target schema.
15. Calls PCM_OP_CUST_UPDATE_SERVICES to close the service on the source schema.
16. Sets the status flag of the source subscription service to PIN_STATUS_FLAG_DUE_TO_SERVICE_TRANSFER, and sets the status to **Closed**.

17. Sets the status flag of the source member services to PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE, and sets the status to **Closed**.
18. Generates an **/event/customer/status** event.
19. Creates an **/event/audit/subscription/transfer** audit event to record the service transfer. The **/event/audit/subscription/transfer** event contains PIN_FLD_TO_SERVICE_OBJ, PIN_FLD_TO_BAL_GRP_OBJ, and PIN_FLD_TO_PROFILE_OBJ.
20. Calls PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION if the **TransferScheduledActions** business parameter is enabled.

If the source and target balance groups of the transferred services are in different schemas, PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER does the following:

1. Calls PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION, which finds all the services and their new service references and uses the references to purchase bundles on the services in PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER.

> **Note:**
>
> For PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER to call PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION:
>
> - Either PIN_FLD_FROM_BAL_GRP_OBJ or the **/service** object must be present in the input flist.
> - Either the **/billinfo** object or the **/billinfo** array must be present in the input flist.

2. Creates an **/event/audit/service_balgrp_transfer** audit event to record the balance group transfer. The **/event/audit/service_balgrp_transfer** audit event contains the PIN_FLD_TO_SERVICE_OBJ field.

Managing Profile Sharing Groups

See the following topics:

- Creating a Profile Sharing Group
- Validating Profile Sharing Group Members
- Modifying a Profile Sharing Group
- Deleting Members and Profiles from a Profile Sharing Group
- Changing the Owner of a Profile Sharing Group
- Deleting a Profile Sharing Group

Creating a Profile Sharing Group

To customize a third-party application to create a profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE.

The profile sharing group has either a service or an account as the owner and a list of member services or accounts. It also contains one or more **/profile** objects owned by the group owner that is shared with group members.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE creates the group object and then adds the members and profiles:

1. Creates a **/group/sharing/profiles** object and assigns the group owner.
2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/profiles** object. Each PIN_FLD_MEMBERS array element specifies an **/account** object and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the profile sharing group. However, subclass instances of the service type do not become members.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/profiles** object is created. The member account can join the group and purchase the service later.
3. Validates that the group owner does not belong to a profile sharing group owned by one of the members.
4. Calls PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS to validate the members. By default, this opcode has no validation rules for profile sharing group members, but you can customize it to validate members. See "Validating Profile Sharing Group Members".

   > **Note:**
   >
   > PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does not validate that the primary currency of members matches the parent.

5. Adds the **/profile** objects in the PIN_FLD_PROFILES array to the PROFILES array of the **/group/sharing/profiles** object.

   The profiles must be owned by the profile sharing group owner and must have a current validity period.

   If the PIN_FLD_PROFILES array is empty, the profiles list is empty. Duplicate entries are ignored.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE returns the POID of the profile sharing group object created and the POID of the event generated to record the creation.

When the profile sharing group is created, the group is automatically added to each member's ordered balance group.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE fails if the owner is a member of a sharing group owned by one of the members.

Validating Profile Sharing Group Members

To validate the members of a profile sharing group, use PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS. By default, this opcode does nothing.

You can customize this opcode to implement your validation rules. For example, the opcode can make sure that a member's service type is the same as the owner's service type.

This opcode takes as input the list of potential members and returns only those members that pass validation.

This opcode is called by PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE and PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY.

If customized for profile sharing groups, PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS validates members before they are added or modified and returns a list of valid members to the calling opcode.

Modifying a Profile Sharing Group

To modify a profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY. This opcode performs one of the following modifications, depending on the input POID:

- Adds member accounts or services to an existing group.
- Adds profiles to an existing group.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY returns the POID of the group that was modified and the POIDs of the events that were generated to record the group modification.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY fails in the following cases:

- If the owner is a member of its own group or a group owned by a group member.
- If a profile object in the input flist is not valid.

For more information, see:

- Adding Members to a Profile Sharing Group
- Adding Profiles to a Profile Sharing Group

Adding Members to a Profile Sharing Group

To add members to a profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY.

> **Note:**
>
> The element ID for each member in the input flist must be unique within the membership. If an element ID is already being used by an existing member in the group object you are modifying, the opcode overwrites the existing member.
>
> To make sure that you do not assign a new member to an existing member's ID, include all the existing members in the input flist. In this case, the opcode will make sure that all new members are added, even if a new member's element ID is the same as an existing member's ID.
>
> This opcode handles duplicate members, so including existing members in the input flist does not cause problems.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/profiles** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** object and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   > **Note:**
   >
   > - The guidelines for creating a profile sharing group also apply to adding members to an existing group.
   > - If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/profiles** object is modified. The member account can join the group and purchase the service later.

2. Validates that the group owner does not belong to a profile sharing group owned by one of the members.
3. Calls PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS to validate the members. By default, this opcode has no validation rules for profile sharing group members, but you can customize it to validate members. See "Validating Profile Sharing Group Members".
4. Creates an flist that includes the list of members to be added to the sharing group.
5. Generates an **/event/group/sharing/profiles/modify** event to record the changes.

Adding a member to a profile sharing group triggers PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS, which adds the profile sharing group to the ordered balance group for each member.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS creates or modifies ordered balance groups for profile sharing group members (accounts or services) when the profile sharing group is created or modified.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS is triggered when a profile sharing group is created or modified. This opcode calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY to create or modify ordered balance groups.

You can customize PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS to use it with other types of sharing groups or to use it with only certain profile sharing groups.

This opcode is listed in the **pin_notify** file for these events:

- **/event/group/sharing/profiles/create**
- **/event/group/sharing/profiles/modify**

When merging event notification files, list PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS after the Enterprise Application Integration (EAI) framework publishing opcode, PCM_OP_PUBLISH_GEN_PAYLOAD. See the discussion on merging event notification lists in *BRM Developer's Guide*.

Use PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE to create an ordered balance group for a new service purchased by an existing account, if that service automatically belongs to a sharing group. By default, this opcode creates an ordered balance group for a service that is a member of a profile sharing group.

A newly purchased service belongs to a profile sharing group if its service type already belongs to a profile sharing group and the group is defined to include services added in the future.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE is triggered when a new service is added to an existing account.

You can customize this opcode to use it with other types of sharing groups or to use it with only certain profile sharing groups.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE is listed in the **pin_notify** file for **/event/notification/service/create**. When merging event notification files, this opcode should be listed after PCM_OP_TCF_PROV_CREATE_SVC_ORDER. See the discussion on merging event notification lists in *BRM Developer's Guide*.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE is not called by any opcode.

Adding Profiles to a Profile Sharing Group

When adding profiles to a profile sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/profile** objects in the PIN_FLD_PROFILES array to the PROFILES array of the **/group/sharing/profiles** object.
2. Validates that the new group name is not a duplicate of an existing group name.
3. Creates an flist that includes the list of profiles to be added to the sharing group.
4. Generates an **/event/group/sharing/profiles/modify** event to record the changes.

Deleting Members and Profiles from a Profile Sharing Group

> **Note:**
>
> The element ID for each member in the **/group/sharing/profiles** object is unique. If you use an incorrect element ID for the member you want to delete, the opcode deletes the wrong member.

To delete a member or profile from a profile sharing group, your application calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY to list the members or profiles to be deleted. This opcode passes an empty array with an element ID for each listed object to be deleted.

> **Note:**
>
> If the array passed by the opcode is not empty, you are either adding or modifying the member or profile.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY then calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE.

To delete a profile sharing group member, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **group/sharing/profiles** object from the member's **/ordered_balgrp** object.
2. Deletes the member from the **/group/sharing/profiles** members array.
3. Generates an **/event/group/sharing/profiles/delete** event.

To delete a shared profile from the profile sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Deletes the **/profile** objects specified in the PIN_FLD_PROFILES array from the PROFILES array of the **/group/sharing/profiles** object.
2. Generates an **/event/group/sharing/profiles/delete** event to record the deletion.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE returns the POID of the profile sharing group object that was modified and the POID of the event that was generated.

Changing the Owner of a Profile Sharing Group

To change the owner of an existing profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT.

This opcode takes the following fields as input:

- PIN_FLD_GROUP_OBJ: The profile sharing group (**/group/sharing/profiles**) whose owner is being changed.
- PIN_FLD_POID: The **/account** object of the current owner.
- PIN_FLD_PARENT: The new owner. This field identifies the **/service** object designated as the new owner of the profile sharing group. The **/service** object can be in the account that currently owns the profile sharing group or it can be in a different account.
- PIN_FLD_ACCOUNT_OBJ: The **/account** object of the new owner.

> **Note:**
>
> If you are changing the ownership of the profile sharing group from one service to another in the same account, the object passed in this field matches the one in the PIN_FLD_POID field.

- PIN_FLD_BAL_GRP_OBJ: The balance group used to track sharing activities for the new owner.
- PIN_FLD_PROFILES: An array of profiles that the new owner will share with the group members. This field is optional but should be included if the shared profiles for the new owning service are different than those for the current owning service.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - The input flist contains the **/balance_group** object and **/account** object for the new profile sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - There are no cyclical relationships. For example, the owner cannot be a member of its own group. Also, if a member of the group owns a profile sharing group, the owner cannot be a member of that group.

2. Deletes the current owner's profiles from the **/group/sharing/profiles** object.
3. Adds the list of profile instances of the new parent passed in the input flist to the group object.
4. Sets the **/group/sharing/profile** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
5. Adds the new owner's shared profiles to the **/group/sharing/profiles** object.
6. Generates an **/event/group/sharing/profiles/modify** event to record the owner change.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT returns the POID of the profile sharing group that was modified and the event that was generated to record the owner change.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT fails if the new owner that is passed is already a member of the profile sharing group.

Deleting a Profile Sharing Group

To delete a profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE.

If the member and profiles arrays are empty in the input flist, the opcode assumes you are deleting the group.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **group/sharing/profiles** object from each member's **/ordered_balgrp** object.
2. Deletes the **group/sharing/profiles** object.
3. Generates an **/event/group/sharing/profiles/delete** event to record the deletion.

Using Access Control Lists

See the following topics:

- Managing ACL Groups
- Accessing /group/acl Data

Managing ACL Groups

BRM stores information about each ACL in **/group/acl** objects in the BRM database. You manage or access data in **/group/acl** objects by using the Permissioning facilities module (FM) standard opcodes.

You create, modify, and delete **/group/acl** objects by using PCM_OP_PERM_ACL* opcodes. These opcodes validate data in the input flist and then call Group FM standard opcodes to create, add, modify, or delete data in a **/group/acl** object. For example, PCM_OP_PERM_ACL_GROUP_ADD_MEMBER adds members to a **/group/acl** object by calling PCM_OP_GROUP_ADD_MEMBER.

To manage **/group/acl** objects, you use the following Permissioning FM standard opcodes:

- To create a **/group/acl** object, use PCM_OP_PERM_ACL_GROUP_CREATE.
- To add a member to a **/group/acl** object, use PCM_OP_PERM_ACL_GROUP_ADD_MEMBER.

- To delete a member from a **/group/acl** object, use PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER.
- To modify the attributes in a **/group/acl** object, such as the ACL name, group account, or list of authorized service types, use PCM_OP_PERM_ACL_GROUP_MODIFY.

> **Note:**
>
> PCM_OP_PERM_ACL_GROUP_MODIFY overwrites the list of authorized services in the **/group/acl** object. When the PIN_FLD_PERMITTEDS array is included in the input flist, PCM_OP_PERM_ACL_GROUP_MODIFY deletes all services from the **/group/acl** object and adds the services from the input flist.

- To delete an entire **/group/acl** object, use PCM_OP_PERM_ACL_GROUP_DELETE.

> **Note:**
>
> Deleting an ACL does not affect services. It only removes the ACL.

Accessing /group/acl Data

Use the following opcodes to access data in **/group/acl** opcodes. Client applications typically use these opcodes to find all ACLs that a CSR belongs to or to determine which account group to display.

- To find all ACLs to which a CSR or member belongs, use PCM_OP_PERM_FIND. See "Finding CSR Membership".
- To determine which account group to display in a client application, use PCM_OP_PERM_GET_CREDENTIALS and PCM_OP_PERM_SET_CREDENTIALS.

Finding CSR Membership

Use PCM_OP_PERM_FIND to find all ACLs to which a CSR belongs. This opcode returns a list of all groups that a CSR is authorized to access, along with specific information about each ACL.

Customizing the Account Dump Utility (ADU)

Use the account dump policy opcodes to customize Account Dump Utility (ADU) validation and output file format. The Account Dump utility (ADU) is a diagnostics tool that enables you to validate account information before or after certain business processes (for example, after completion of a migration or upgrade or before billing or payment allocation).

Use PCM_OP_ADU_POL_DUMP to convert the ADU output format to a format other than XML. By default, PCM_OP_ADU_VALIDATE dumps account information in XML format. PCM_OP_ADU_POL_DUMP allows you to convert the account information in the input flist into a format other than XML (for example, CSV).

Use PCM_OP_ADU_POL_VALIDATE to define custom validations for validating account data. By default, PCM_OP_ADU_VALIDATE performs the predefined validations enabled in the ADU **pin.conf** file. PCM_OP_ADU_POL_VALIDATE allows you to define custom validations for validating the account information provided in the input flist.

Any custom validation you define must be associated with a validation code. For PCM_OP_ADU_VALIDATE to perform the custom validations, they must be configured and enabled in the ADU pin.conf.

Business Profile Opcode Workflows

See the following topics:

- Creating Business Profiles
- Assigning Bill Units to Business Profiles
- Getting Information about a Business Profile

Creating Business Profiles

Use PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE to create business profiles.

PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE creates one **/associated_bus_profile** object for each bill unit in an account.

If the BRM-Oracle Analytics Publisher invoice integration is enabled, during customer account creation, PCM_OP_CUST_CREATE_BILLINFO calls PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE to create one **/associated_bus_profile** object for each bill unit in the account.

Assigning Bill Units to Business Profiles

You can assign a bill unit to a business profile (**/config/business_profile** object) during or after account creation.

To assign a bill unit to a business profile during account creation:

1. Call PCM_OP_CUST_COMMIT_CUSTOMER.

   In the opcode's input flist, specify the **/config/business_profile** object's POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the appropriate BILLINFO array element.
2. PCM_OP_CUST_COMMIT_CUSTOMER passes the business profile information to an internal opcode.
3. BRM calls the following opcodes to validate the bill unit and its balance groups and services against the rules in the validation templates associated with the business profile:
   - PCM_OP_CUST_SET_BILLINFO, which validates the **/billinfo** object against the bill unit validation template (**/config/template/billinfo** object) and sets the **/billinfo** object's PIN_FLD_OBJECT_CACHE_TYPE value.
   - PCM_OP_CUST_SET_BAL_GRP, which validates the balance group objects against the balance group validation template (**/config/template/balance_group** object) and sets the **/balance_group** object's PIN_FLD_OBJECT_CACHE_TYPE value. When a service is associated with a balance group, this opcode also validates the service object against the service validation template that corresponds to the service object type (**/config/template/service** or **/config/template/service/***) and sets the **/service** object's PIN_FLD_OBJECT_CACHE_TYPE value.

> **Note:**
>
> If the business profile is not associated with a validation template that matches a particular object type, validation is not performed on the corresponding object type. Skipping validation for this reason is not equivalent to validation failure and does not prevent the bill unit from being assigned to the business profile.

4. One of the following results occurs:
   - *If no validation errors occur,* BRM calls PCM_OP_CUST_SET_BILLINFO to put the business profile POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the new **/billinfo** object.
   - *If validation errors occur,* account and bill unit creation fail.

Whenever invoice business profiles are modified in the **/config/business_profile** object, use PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE to update all related / associated_bus_profile objects.

To assign a bill unit to a business profile after account creation, see "Changing a Bill Unit's Business Profile".

Changing a Bill Unit's Business Profile

Use PCM_OP_CUST_CHANGE_BUSINESS_PROFILE to perform these operations:

- Assign a bill unit to a business profile after an account has been created.
- Change the business profile of a bill unit that currently belongs to a different business profile.

Include the following information in PCM_OP_CUST_CHANGE_BUSINESS_PROFILE input flist:

- The POID of the **/billinfo** object whose business profile you want to change.
- The POID of the new **/config/business_profile** object to assign the **/billinfo** object to.
- The POID of any associated **/balance_group** objects that must be modified for the new business profile *and* the required modifications.
- The POID of any associated **/service/*** objects that must be modified for the new business profile *and* the required modifications.

PCM_OP_CUST_CHANGE_BUSINESS_PROFILE performs these operations:

1. Validates the **/billinfo** object against the bill unit validation template (**/config/template/billinfo** object) associated with the new business profile and then does one of the following:

   If validation succeeds:

   a. Calls PCM_OP_CUST_SET_BILLINFO to change the **/config/business_profile** POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the **/billinfo** object.

   b. If cache residency distinction is enabled, changes the **/account** object's PIN_FLD_OBJECT_CACHE_TYPE value if required, then validates all cache residency objects that are related to the account and are in the account context (for example, **/profile/acct_extrating** objects and account **/purchased_product** objects).

   If validation fails, returns an error.

2. Creates a list of all the objects associated with the bill unit.
3. For each balance group in the list, checks whether the input flist contains a requested modification.

   If a modification is requested, calls PCM_OP_CUST_SET_BAL_GRP to make the modifications and to validate the modified balance group object against the balance group validation template (**/config/template/balance_group** object) associated with the new business profile.

   If a modification is *not* requested, validates the balance group itself.
4. After validating a balance group, creates a list of all the services associated with the balance group.
5. If cache residency distinction is enabled, searches for objects related to each service in the list and validates them (for example, **/purchased_discount** and **/profile/serv_extrating** objects that belong to the account).
6. For each service in the list, checks whether the input flist contains a requested modification.

   If a modification is requested, calls PCM_OP_CUST_MODIFY_SERVICE to make the modifications and to validate the modified service object against the service validation template that most closely corresponds to the service object type (**/config/template/service** or **/config/template/service/*** object) associated with the new business profile.

   If a modification is *not* requested, validates the service itself.
7. Repeats steps 3 through 6 until all balance groups and services associated with the bill unit are validated.

   If any of the balance group or service validations fail, returns an error. If an error is returned, the business profile change initiated in step 1 does not take effect.

> **Note:**
>
> If the business profile is not associated with a validation template that matches a particular object type, validation is not performed on the corresponding object type. Skipping validation for this reason is not equivalent to validation failure and does not prevent the bill unit from being assigned to the business profile.

Getting Information about a Business Profile

A business profile (**/config/business_profile** object) and the validation templates (**/config/template** subclass objects) linked to it include key-value pairs in their PIN_FLD_PAIR arrays. To get information about the objects belonging to a business profile, you can retrieve the values of the keys in the business profile and its associated validation templates. For example, if a bill unit belongs to a business profile that has a PIN_FLD_PAIR_KEY field set to **IsPrepaidGold** and a corresponding PIN_FLD_PAIR_VALUE field set to **Yes**, you know that the charges associated with the bill unit are prepaid with a gold credit card.

To get the value of a key in the business profile and validation templates associated with an object, call PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO. Include the following information in the opcode's input flist:

- The POID of the **/config/business_profile** object to which the related object belongs. These are the possible related objects:

  - **/balance_group**
  - **/billinfo**
  - **/group/sharing/charges**
  - **/group/sharing/discounts**
  - **/group/sharing/profiles**
  - **/ordered_balgrp**
  - **/profile/acct_extrating**
  - **/profile/serv_extrating**
  - **/purchased_discount**
  - **/purchased_product**
  - **/service/***
- The key whose value you want the opcode to get.
- The type of object from which the key's value should be obtained. These are the possible object types:
  - **/config/business_profile**
  - **/config/template/balance_group**
  - **/config/template/billinfo**
  - **/config/template/group**
  - **/config/template/sharing**
  - **/config/template/ordered_balgrp**
  - **/config/template/profile/***
  - **/config/template/purchased_discount**
  - **/config/template/purchased_product**
  - **/config/template/service/***

Creating and Managing Account Hierarchies

To create and manage account hierarchies, BRM uses the following PCM_OP_BILL_GROUP_* opcodes:

> **Note:**
>
> These opcodes call other PCM_OP_GROUP_* opcodes to perform functionality. Your custom client application should *not* call the PCM_OP_GROUP_* opcodes.

- PCM_OP_BILL_GROUP_CREATE

  See "Creating Account Hierarchies".
- PCM_OP_BILL_GROUP_ADD_MEMBER

  See "Adding Accounts to Account Hierarchies".
- PCM_OP_BILL_GROUP_MOVE_MEMBER

See "Moving Accounts from One Account Hierarchy to Another".

- PCM_OP_BILL_GROUP_DELETE_MEMBER

  See "Deleting Accounts from Account Hierarchies".

- PCM_OP_BILL_GROUP_DELETE

  See "Deleting Account Hierarchies".

- PCM_OP_BILL_GROUP_GET_PARENT

  See "Finding the Parent of an Account Hierarchy".

- PCM_OP_BILL_GROUP_GET_CHILDREN

  See "Getting a List of Child Accounts in an Account Hierarchy".

Creating Account Hierarchies

To create an account hierarchy, BRM uses PCM_OP_BILL_GROUP_CREATE.

This opcode creates a **/group** object for the account hierarchy.

The input to PCM_OP_BILL_GROUP_CREATE is the following:

- The name of the **/group** object
- The parent **/account** object to own the group

PCM_OP_BILL_GROUP_CREATE calls PCM_OP_GROUP_CREATE_GROUP to create the group. The new group's POID is written to the PIN_FLD_GROUP_OBJ field in the specified **/account** object. The output of PCM_OP_GROUP_CREATE_GROUP is returned as the output of this call.

This operation is performed inside a transaction.

The PCM_OP_BILL_GROUP_CREATE output flist includes the following:

- The POID of the group that was created
- A PIN_FLD_RESULTS array with a single event element

The results array returns the POID of an **/event** object that holds the old and new value of the parent object. Because an object was created and an old value did not exist, the old value is returned as NULL. The old and new parent values are kept in the EVENT_GROUP_PARENTS_T table, which is linked to the EVENT_T table.

If an error occurs, a NULL flist is returned.

Adding Accounts to Account Hierarchies

To add an account to an account hierarchy, BRM uses PCM_OP_BILL_GROUP_ADD_MEMBER.

The input to PCM_OP_BILL_GROUP_ADD_MEMBER includes the following:

- The **/group** object to add accounts to
- A set of POIDs of the accounts to add

Any account can be added to an account hierarchy, including the parent account of another account hierarchy.

> **Note:**
>
> Each child account can belong to only one parent account.

PCM_OP_BILL_GROUP_ADD_MEMBER calls PCM_OP_GROUP_ADD_MEMBER to add the accounts. After the accounts are successfully added, an **/event** object is created that contains a list of the added accounts. The POID of the new **/event** object is returned.

Only accounts not already in the group are added. If the list of accounts to add contains accounts already in the account group, they are ignored.

When an account is added to the group, its bill time information (the NEXT_BILL_T field in the **/billinfo** object) will be updated to match that of the parent's **/billinfo** object.

> **Note:**
>
> If you want to separate the billing cycles for the child account:
>
> 1. Create the service/child account as a standalone account.
> 2. Run billing for all previous cycles to establish the desired billing history for the standalone account.
> 3. After the billing history is established, move the account under the hierarchy.

PCM_OP_BILL_GROUP_ADD_MEMBER is performed inside a transaction.

The PCM_OP_BILL_GROUP_ADD_MEMBER output flist includes the following:

- The POID of the modified group
- A PIN_FLD_RESULTS array with a single event element

The results array returns the POID of an **/event** object that holds the list of accounts added to the group. The list of added accounts is kept in the EVENT_GROUP_MEMBERS_T table, which is linked to the EVENT_T table.

Moving Accounts from One Account Hierarchy to Another

To move an account from one account hierarchy to another, BRM uses PCM_OP_BILL_GROUP_MOVE_MEMBER.

This opcode is the recommended way to perform this action. It is a wrapper for the other PCM_OP_BILL_GROUP opcodes.

When an account is moved, PCM_OP_BILL_GROUP_MOVE_MEMBER calls PCM_OP_BILL_GROUP_DELETE. If the account being moved is the last account in a group, PCM_OP_BILL_GROUP_MOVE_MEMBER also calls PCM_OP_BILL_GROUP_DELETE to remove the group.

The PIN_FLD_FLAGS field in PCM_OP_BILL_GROUP_MOVE_MEMBER input flist controls the type of information returned in the output flist. This field can have the following values:

- **1:** If a bill unit of the account to be moved is a member of a collections sharing group, the opcode returns the details of the collections sharing group. When the flag is set to this value, the account is not moved.

- **2:** If a bill unit of the account to be moved is a member of a collections sharing group, the opcode calls PCM_OP_COLLECTIONS_DELETE_GROUP_MEMBER to delete the bill unit from the collections sharing group and then continues moving the account to another account hierarchy.

If the account is moving into an existing account hierarchy, PCM_OP_BILL_GROUP_MOVE_MEMBER also calls PCM_OP_BILL_GROUP_ADD_MEMBER to add the account to the target group.

If the account is moving into a group that does not exist, PCM_OP_BILL_GROUP_MOVE_MEMBER calls PCM_OP_BILL_GROUP_CREATE to create the group.

If the target account (the top-level parent account of the nonexistent target group) also does not exist, PCM_OP_BILL_GROUP_MOVE_MEMBER attempts to remove the moving account from its group and leave it as a standalone account. If the moving account has a nonpaying bill unit, nothing is done, and an error is returned.

Deleting Accounts from Account Hierarchies

To delete an account from an account hierarchy, BRM uses PCM_OP_BILL_GROUP_DELETE_MEMBER.

PCM_OP_BILL_GROUP_DELETE_MEMBER calls PCM_OP_GROUP_DELETE_MEMBER to perform the deletion. After the accounts are successfully deleted from the group, an **/event** object is created that contains a list of the deleted accounts. The POID of the **/event** object is returned.

The list of accounts to delete can contain POIDs of accounts that are not in the account group; those accounts are ignored. Only accounts in the group are deleted from the account group.

You cannot delete an account if the bill unit it owns is a paying bill unit of a nonpaying bill unit in another account.

Deleting Account Hierarchies

To delete an account hierarchy, BRM uses PCM_OP_BILL_GROUP_DELETE.

This opcode performs the following actions:

- Deletes the specified **/group** object
- Deletes the accounts list for that account group
- Clears the PIN_FLD_GROUP_OBJ field in the group parent's **/account** object

PCM_OP_BILL_GROUP_DELETE calls PCM_OP_GROUP_DELETE_GROUP. The input to PCM_OP_BILL_GROUP_DELETE is the POID of the group to delete. After successfully deleting the group, the opcode returns the deleted group's POID.

Finding the Parent of an Account Hierarchy

To find the parent of an account hierarchy, use PCM_OP_BILL_GROUP_GET_PARENT.

PCM_OP_BILL_GROUP_GET_PARENT retrieves the parent account of a specified **/group** object. The input to PCM_OP_BILL_GROUP_GET_PARENT is an account hierarchy POID. The account POID identifying the group's parent account is returned.

Getting a List of Child Accounts in an Account Hierarchy

To get a list of child accounts in an account hierarchy, use PCM_OP_BILL_GROUP_GET_CHILDREN.

To read only specified **/account** fields for each account (for example, account name), pass the specified fields in the input flist along with the POID of the **/group** object. If the input flist contains only the **/group** object POID, all the fields in the ACCOUNT table for each child are returned.

About the PCM_OP_GROUP Opcodes

The Group FM includes the following opcodes. These opcodes are called by other opcodes (for example, the wrapper opcode PCM_OP_BILL_GROUP_CREATE calls PCM_OP_GROUP_CREATE_GROUP to create a group). Always use the wrapper opcodes, such as the PCM_OP_BILL wrapper opcodes, not the PCM_OP_GROUP opcodes.

- PCM_OP_GROUP_CREATE_GROUP

  See "Creating Groups".
- PCM_OP_GROUP_ADD_MEMBER

  See "Adding Members to Groups".
- PCM_OP_GROUP_DELETE_MEMBER

  See "Deleting Members from Groups".
- PCM_OP_GROUP_SET_PARENT

  See "Setting a Group Parent".
- PCM_OP_GROUP_DELETE_GROUP

  See "Deleting Groups".
- PCM_OP_GROUP_UPDATE_INHERITED

  See "Updating Inheritance Fields in Groups".

Creating Groups

To create a **/group** object, BRM uses PCM_OP_GROUP_CREATE_GROUP.

PCM_OP_GROUP_CREATE_GROUP first initializes the new group. Then this opcode uses PCM_OP_GROUP_SET_PARENT to set any parent information specified on input. If a members list is passed in, PCM_OP_GROUP_ADD_MEMBER is called to add those members.

PCM_OP_GROUP_CREATE_GROUP does not generate an **/event** object. However, if new members are specified and added to the group, that operation generates an **/event** object. The output of PCM_OP_GROUP_SET_PARENT and PCM_OP_GROUP_ADD_MEMBER are attached to the return flist.

Adding Members to Groups

To add members to a group, BRM uses PCM_OP_GROUP_ADD_MEMBER.

The input to this opcode is a set of POIDs of the members to add. After the members are successfully added, an **/event** object is created containing a list of the added members. The POID of the **/event** object is returned.

The add members list might contain POIDs of members already in the group. Those members are ignored; only new members are added to the group.

Deleting Members from Groups

To delete members from a group, BRM uses PCM_OP_GROUP_DELETE_MEMBER.

After the members are deleted, an **/event** object is created containing a list of the deleted members. The POID of the **/event** object is returned.

The delete members list might contain POIDs of members not in the group. Those elements are ignored; only existing members are deleted from the group.

Setting a Group Parent

To set the parent object of a group, BRM uses PCM_OP_GROUP_SET_PARENT.

The parent field is specified using a POID ID, which is similar to a member element of a group. However, a parent is considered the object that owns the group. If the group represents accounts, the members elements refer to accounts that belong to the group.

The results array returns the POID of an **/event** object that holds the old and new value of the parent object.

Deleting Groups

To delete a **/group** object, BRM uses PCM_OP_GROUP_DELETE_GROUP.

PCM_OP_GROUP_DELETE_GROUP removes a specified **/group** object and all members linked to it.

Updating Inheritance Fields in Groups

To update the inheritance fields of a group, BRM uses PCM_OP_GROUP_UPDATE_INHERITED.

The input to PCM_OP_GROUP_UPDATE_INHERITED is the POID of the group to update followed by the inheritance information. The inheritance information is passed as a substructure and must have space allocated for it as a separate object, independent of the **/group** object.

10

Deposit Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) deposit opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Creating Deposit Specification Profiles
- Validating Deposit Specification Profiles
- Getting Deposit Specification Profiles
- Modifying Deposit Specification Profiles
- Creating Deposit Specifications
- Validating Deposit Specifications
- Getting Deposit Specifications
- Modifying Deposit Specifications
- Deleting Draft Deposit Specifications
- Purchasing Deposits
- Validating a Deposit Purchase
- Getting Deposits
- Updating Deposits
- Validating Deposit Updates
- Collecting Deposit Payments
- Validating Deposit Payment Collections
- Reversing Deposits
- Validating a Deposit Reversal
- Releasing Deposits
- Validating a Deposit Release
- Transferring Deposits
- Validating a Deposit Transfer
- Adding Interest to a Deposit
- Customizing the Interest Calculation
- Requesting Deposit Refunds
- Validating a Deposit Refund Request
- Getting Refund Requests
- Updating Refund Requests
- Updating Received Deposits

- Triggering Deposits
- Retrieving Transaction Details of a Deposit

Opcodes Described in This Chapter

Table 10-1 lists the deposit management opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 10-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE	Creating Deposit Specification Profiles Modifying Deposit Specification Profiles Validating Deposit Specification Profiles
PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE	Validating Deposit Specification Profiles
PCM_OP_DEPOSIT_GET_SPECIFICATION_PROFILE	Getting Deposit Specification Profiles
PCM_OP_DEPOSIT_MODIFY_SPECIFICATION_PROFILE	Modifying Deposit Specification Profiles Validating Deposit Specification Profiles Creating Deposit Specification Profiles
PCM_OP_DEPOSIT_CREATE_SPECIFICATION	Creating Deposit Specifications Validating Deposit Specifications
PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION	Validating Deposit Specifications
PCM_OP_DEPOSIT_GET_SPECIFICATION	Getting Deposit Specifications
PCM_OP_DEPOSIT_MODIFY_SPECIFICATION	Modifying Deposit Specifications Validating Deposit Specifications
PCM_OP_DEPOSIT_PURCHASE_DEPOSIT	Purchasing Deposits Validating a Deposit Purchase
PCM_OP_DEPOSIT_POL_VALID_PURCHASE_DEPOSIT	Validating a Deposit Purchase
PCM_OP_DEPOSIT_GET_DEPOSITS	Getting Deposits
PCM_OP_DEPOSIT_UPDATE_DEPOSIT	Updating Deposits
PCM_OP_DEPOSIT_POL_VALID_UPDATE_DEPOSIT	Validating Deposit Updates
PCM_OP_DEPOSIT_COLLECT_PAYMENT	Collecting Deposit Payments
PCM_OP_DEPOSIT_POL_VALID_COLLECT_PAYMENT	Validating Deposit Payment Collections
PCM_OP_DEPOSIT_REVERSE_DEPOSIT	Reversing Deposits Validating a Deposit Reversal

Table 10-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT	Validating a Deposit Reversal
PCM_OP_DEPOSIT_RELEASE_DEPOSIT	Releasing Deposits Validating a Deposit Release
PCM_OP_DEPOSIT_POL_VALID_RELEASE_DEPOSIT	Validating a Deposit Release
PCM_OP_DEPOSIT_TRANSFER_DEPOSIT	Transferring Deposits Validating a Deposit Transfer
PCM_OP_DEPOSIT_POL_VALID_TRANSFER_DEPOSIT	Validating a Deposit Transfer
PCM_OP_DEPOSIT_ADD_INTEREST	Adding Interest to a Deposit
PCM_OP_DEPOSIT_POL_PRE_ ADD_INTEREST PCM_OP_DEPOSIT_POL_POST_ADD_INTEREST	Customizing the Interest Calculation
PCM_OP_DEPOSIT_REFUND_REQUEST	Requesting Deposit Refunds
PCM_OP_DEPOSIT_POL_VALID_REFUND_REQUEST	Validating a Deposit Refund Request
PCM_OP_DEPOSIT_GET_REFUND_REQUEST	Getting Refund Requests
PCM_OP_DEPOSIT_UPDATE_REFUND_REQUEST	Updating Refund Requests
PCM_OP_DEPOSIT_UPDATE_DEPOSIT_RECEIVED	Updating Received Deposits
PCM_OP_DEPOSIT_TRIGGER_DEPOSIT	Triggering Deposits

Creating Deposit Specification Profiles

Use the PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE opcode to create a deposit specification profile. Profiles define common business rules for deposits and are used to create deposit specifications.

In the input flist, you can set the profile name, description, start date, end date, interest calculation flags, transferable flags, interest type, interest rate, interest offset details, and details about handling refund fees.

This opcode creates a **/config/deposit_spec_profile** object. This opcode is called by Billing Care.

The PCM_OP_DEPOST_CREATE_SPECIFICATION_PROFILE opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE opcode to validate the input parameters.
- Checks the results of PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE and sends appropriate error messages.
- Searches the database for the **/config/deposit_spec_profile** with the name. If it already exists, sends an error.
- Calls the PCM_OP_CREATE_OBJ opcode to create the **/config/deposit_spec_profile** object.

Validating Deposit Specification Profiles

Use the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE policy opcode to validate and implement any customizations while creating deposit specification profile information. It validates if the deposit specification profile name is provided. This policy opcode is called by the PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE opcode.

PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE validates the deposit specification profile information in the **/config/deposit_spec_profile** object. PCM_OP_DEPOSIT_CREATE_SPECIFICATION_PROFILE does the following before calling PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE:

- Validates that the deposit specification profile mandatory parameters are in the input.
- Calls the PCM_OP_SEARCH opcode with the provided deposit specification profile name. If the profile exists, it returns an error.
- Validates that the details of PIN_FLD_DEPOSIT_PROFILE_SPEC in the input flist use the required format.
- Validates if the PIN_FLD_INTEREST_TYPE is provided. There are three interest types: simple, compound, and absolute. If the type is not present, sends an error.
- If any BRM-related error occurs, sets the PIN_FLD_RESULT field to FAIL, shares a designated error message to the output flist, and clears the error buffer. If there are no errors, reviews all related areas. If there are no failures, sets the PIN_FLD_RESULT field to PASS.

Getting Deposit Specification Profiles

Use the PCM_OP_DEPOSIT_GET_SPECIFICATION_PROFILE opcode to get any existing deposit specification profiles available in the BRM database using its name or POID. This opcode is called by Billing Care.

PCM_OP_DEPOSIT_GET_SPECIFICATION_PROFILE does the following:

- Validates the mandatory parameters in the input flist.
- Prepares the search template to search the **/config/deposit_spec_profile** object.
- Calls the PCM_OP_SEARCH opcode to find the **/config/deposit_spec_profile** object and prepares the output flist.

Modifying Deposit Specification Profiles

Use the PCM_OP_DEPOSIT_MODIFY_SPECIFICATION_PROFILE opcode to update, edit, or modify an existing deposit specification profile attributes in the **/config/deposit_spec_profile** object. This opcode is called by Billing Care.

PCM_OP_DEPOSIT_MODIFY_SPECIFICATION_PROFILE does the following:

- Validates the mandatory parameters in the input flist.
- Searches for the given **/config/deposit_spec_profile** object.
- Calls the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE opcode to validate the input parameters.

- Calls the PCM_OP_WRITE_FLDS opcode to update the **/config/deposit_spec_profile** object with the fields provided in the input flist.

Creating Deposit Specifications

Use the PCM_OP_DEPOSIT_CREATE_SPECIFICATION opcode to create deposit specification details. This opcode creates the **/deposit_specification** object.

In the input flist fields, you can set the deposit specification name and select the deposit specification profile, start date, end date, deposit specification code, credit limit flags, draft flag, permitted type and category, and product or plan object.

PCM_OP_DEPOSIT_CREATE_SPECIFICATION does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION opcode to validate if the deposit event or appropriate charge offer is available. If not available, sends an error.
- Searches the database for an existing **/deposit_specification** object with the provided name. If it already exists, sends an error.
- Calls the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE opcode to validate the input parameters.
- Calls the PCM_OP_CREATE_OBJ opcode to create the object.

Validating Deposit Specifications

Use the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION opcode to validate and implement any customizations while creating the deposit specifications. This policy opcode is called by the PCM_OP_DEPOSIT_CREATE_SPECIFICATION opcode.

The PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION policy opcode validates the deposit specification information in the **/deposit_specification** object. Before calling this opcode, PCM_OP_DEPOSIT_CREATE_SPECIFICATION makes the following initial validation checks:

- Validates the deposit specification's mandatory parameters in the input flist.
- Calls the PCM_OP_SEARCH opcode with the provided deposit specification name. If the specification exists, it returns an error.
- Validates that the details of the input flist are in the required format.
- If any BRM-related error occurs, sets the PIN_FLD_RESULT field to FAIL, shares a designated error message to the output flist, and clears the error buffer. If there are no errors, reviews all related areas. If there are no failures, sets the PIN_FLD_RESULT field to PASS.

Getting Deposit Specifications

Use the PCM_OP_DEPOSIT_GET_SPECIFICATION opcode to get existing deposit specification objects. If the name is specified in the input flist, it returns only the particular **/deposit_specification** object, otherwise it returns all objects in the BRM database. This opcode is called by Billing Care.

> **Note:**
>
> By default, the search limit for the opcode is set to 500 and only lists the most recent records. You can customize the search limit according to the requirement.

PCM_OP_DEPOSIT_GET_SPECIFICATION does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_SEARCH opcode to find the specified **/deposit_specification** object and prepares the output flist.

Modifying Deposit Specifications

Use the PCM_OP_DEPOSIT_MODIFY_SPECIFICATION opcode to modify the **/deposit_specification** object attributes. This opcode is called by Billing Care.

PCM_OP_DEPOSIT_MODIFY_SPECIFICATION does the following:

- Validates the mandatory parameters in the input flist.
- Searches for the given **/deposit_specification** with the POID.
- If the specification exists, calls the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION opcode to validate the input parameters.
- Calls the PCM_OP_WRITE_FLDS opcode to update the **/deposit_specification** object with the fields provided in the input flist.

Deleting Draft Deposit Specifications

Use the PCM_OP_DEPOSIT_DELETE_DRAFT opcode to delete draft deposit specifications. The opcode internally validates the status. If the specification is in draft status, the opcode deletes the provided **/deposit_specification** object. Otherwise, the opcode returns an error. This opcode is called by Billing Care.

Purchasing Deposits

Use the PCM_OP_DEPOSIT_PURCHASE_DEPOSIT opcode to create a customer deposit.

Using this opcode, you can select any existing specification from the BRM database and associate it with the customer. The specification must include a deposit charge.

As part of this opcode, a purchase deposit entity is created in the BRM database, the associated charge offer from the deposit specification is purchased by the customer, and the charge is applied. The opcode creates a deposit object that includes all the information about the deposit purchased.

PCM_OP_DEPOSIT_PURCHASE_DEPOSIT triggers the **/deposit_charge** event and the **/item/deposit_charge** item. It creates the **/purchased_deposit** object and makes the payment for the deposit.

> **Note:**
>
> Interest calculation happens only when the payments are received for the account. If the purchased deposit does not collect any payment, the **last_interest_calc_t** field stays empty and is updated in the PCM_OP_DEPOSIT_COLLECT_PAYMENT opcode.

The PCM_OP_DEPOSIT_PURCHASE_DEPOSIT opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_PURCHASE_DEPOSIT opcode to validate the input parameters.
- Calls the PCM_OP_READ_OBJ opcode to get the product details that are configured in the deposit specification.
- Prepares the input flist for the PCM_OP_BILL_MAKE_BILL_NOW opcode to include the **/item/deposit_charge** item.
- Calls the PCM_OP_BAL_GET_BALANCES opcode to get the credit limit.
- Calls the PCM_OP_BILL_SET_LIMIT_AND_CR opcode to increase the credit limit by the deposit amount.
- Gets the **/event/billing/item/transfer** event from the output flist.
- For a payment, including a pre-defined payment, calls the PCM_OP_PYMT_COLLECT opcode to collect the deposit amount.
- Prepares the input flist for the PCM_OP_BILL_ITEM_TRANSFER opcode to allocate the collected or pending payments to the deposit amount.
- Calls the PCM_OP_ACT_USAGE opcode to trigger the **/event/deposit/create** notification event that can be sent to the customer.

Validating a Deposit Purchase

Use the PCM_OP_DEPOSIT_POL_VALID_PURCHASE_DEPOSIT policy opcode to validate and implement any customizations while purchasing a deposit. This policy opcode is called by the PCM_OP_DEPOSIT_PURCHASE_DEPOSIT opcode.

Getting Deposits

Use the PCM_OP_DEPOSIT_GET_DEPOSITS opcode to get the purchased deposits for an account, service, or bill unit from the **/purchased_deposit** object. This opcode is called by Billing Care.

> **Note:**
>
> By default, the search limit for the opcode is set to 500 and only lists the most recent records. You can customize the search limit.

The opcode gets the available deposits associated with a given account or service.

PCM_OP_DEPOSIT_GET_DEPOSITS does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_READ_OBJ opcode if the **/purchased_deposit** object is provided.
- Prepares the search template to search the **/purchased_deposit** object.
- Calls the PCM_OP_SEARCH opcode and prepares the output flist.

Updating Deposits

Use the PCM_OP_DEPOSIT_UPDATE_DEPOSIT opcode to increase or decrease the deposit amount. The opcode reverses the existing **deposit_charge** item and creates a new **deposit_charge** for the given deposit amount and updates the purchased deposit object.

> **Note:**
>
> Requesting an update for a transferred deposit is not valid.

PCM_OP_DEPOSIT_UPDATE_DEPOSIT does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_UPDATE_DEPOSIT opcode to validate the input parameters.
- Calls the PCM_OP_DEPOSIT_REVERSE_DEPOSIT opcode to reverse a deposit.
- Calls the PCM_OP_DEPOSIT_PURCHASE_DEPOSIT opcode to purchase a deposit.
- Creates a custom event to update the deposit and include it in the events array of the **/purchased_deposit** object.

Validating Deposit Updates

Use the PCM_OP_DEPOSIT_POL_VALID_UPDATE_DEPOSIT opcode to validate and implement any customizations while updating the deposit. This policy opcode is called by the PCM_OP_DEPOSIT_UPDATE_DEPOSIT opcode.

Collecting Deposit Payments

Use the PCM_OP_DEPOSIT_COLLECT_PAYMENT opcode to collect pending payments for deposits purchased by customers.

The PCM_OP_DEPOSIT_COLLECT_PAYMENT opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_COLLECT_PAYMENT opcode to validate the input parameters.
- Calls the PCM_OP_READ_OBJ opcode to get the deposit details provided in the input.
- Calls the PCM_OP_PYMT_COLLECT opcode to collect the deposit amount.
- Gets the **/event/billing/item/transfer** event from the output flist.
- Prepares the input flist for the PCM_OP_BILL_ITEM_TRANSFER opcode to transfer the deposit amount.

- Calls the PCM_OP_WRITE_FLDS opcode to update the received deposit amount and the balance amount.
- If the payment received is the first payment, and the calculate interest flag is set, interest calculation starts from the date of this payment.

Validating Deposit Payment Collections

Use the PCM_OP_DEPOSIT_POL_VALID_COLLECT_PAYMENT opcode to validate and implement any customizations while collecting the deposit payments. This policy opcode is called by the PCM_OP_DEPOSIT_COLLECT_PAYMENT opcode.

Reversing Deposits

Use the PCM_OP_DEPOSIT_REVERSE_DEPOSIT opcode to reverse any unintentional or wrongly made deposits. The opcode opens the **/item/payment** object associated with the **/item/deposit_charge** object and creates **/item/deposit/reversal** with the same deposit charge amount.

PCM_OP_DEPOSIT_REVERSE_DEPOSIT does the following:

- Validates the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT opcode with the input flist.
- Gets the PIN_FLD_RESULT from the PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT output flist.
- Searches for **/item/deposit_charge** item for the given **/purchased_deposit**.
- Gets the status and the item total from the **/item/deposit_charge** item.
- If the calculate interest flag is set:
  - Prepares the input flist and calls the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate and add the interest.
  - Gets the interest amount from the PIN_FLD_DEPOSIT_INTEREST_AMOUNT from **/purchased_deposit**.
  - Prepares the input flist for the PCM_OP_PYMT_COLLECT opcode, to provide the interest accumulated as an **/item/payment** object with the G/L ID.
- Calls the PCM_OP_PYMT_COLLECT opcode to create the **/item/payment** object for the interest amount.
- Prepares the input flist for PCM_OP_ACT_USAGE.
- Calls the PCM_OP_BILL_ITEM_TRANSFER to transfer **/item/deposit/reversal** to **/item/deposit_charge**.
- Calls the PCM_OP_BILL_SET_LIMIT_AND_CR opcode to adjust the credit limit amount.
- Calls the PCM_OP_WRITE_FLDS opcode to update the **/purchased_deposit** object.
- Calls the PCM_OP_ACT_USAGE opcode to trigger the **/event/deposit/reversal** notification event that can be sent to the customer.

Validating a Deposit Reversal

Use the PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT opcode to validate and implement any customizations while reversing the deposit. This policy opcode is called by the PCM_OP_DEPOSIT_REVERSE_DEPOSIT opcode.

Releasing Deposits

Use the PCM_OP_DEPOSIT_RELEASE_DEPOSIT opcode to release an existing deposit amount.

You can release the deposit amount as a refund, as a prepayment for due bills or future bills, or to settle the customer's account balance, an option called zeroize.

You can select one of the following modes to release the deposit:

- **Prepayment:** The **/item/payment** object is created immediately.
- **Refund:** The **/deposit_refund_request** object is created and can be approved by the approver. After it is approved, the **/item/payment** and **/item/refund** objects are created and refunded.
- **Zeroize:** The **/item/payment** object is created immediately, the pending items are billed, **/item/payment** is allocated to the bill items, and the remaining balance is refunded to the customer.

PCM_OP_DEPOSIT_RELEASE_DEPOSIT does the following:

- Validates the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_RELEASE_DEPOSIT opcode to validate the input parameters.
- Calls the PCM_OP_WRITE_FLDS opcode to update the **/purchased_deposit** object for the balance amount and amount released.
- Calls the PCM_OP_BILL_MAKE_BILL_NOW opcode to bill the pending items for a specific service or bill unit from the **/purchased_deposit** object.
- If the calculate interest flag is set:
  - Prepares the input flist and calls the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate and add the interest.

    > **Note:**
    >
    > Interest is released only when the complete balance amount is refunded or released. Otherwise, interest is calculated but not released.

  - Gets the interest amount from the PIN_FLD_DEPOSIT_INTEREST_AMOUNT field in the **/purchased_deposit** object.
  - Prepares the input flist for the PCM_OP_PYMT_COLLECT opcode to provide the accumulated interest as **/item/payment**, with the G/L ID.
- Prepares the input flist to call the PCM_OP_BILL_ITEM_TRANSFER opcode, to transfer the deposit amount to a bill.

- Calls the PCM_OP_WRITE_FLDS opcode to update the **/purchased_deposit** object for the balance amount and released amount.
- Calls the PCM_OP_DEPOSIT_REFUND_REQUEST opcode to create the **/deposit_refund_request** object for the deposit amount.
- Calls the PCM_OP_DEPOSIT_UPDATE_REFUND opcode to create the **/item/refund** object.
- Calls the PCM_OP_CUST_SET_NOTE opcode to create the **/note** object.
- Calls the PCM_OP_ACT_USAGE opcode to trigger the **/event/deposit/release** notification event that can be sent to the customer.
- Calls the PCM_OP_WRITE_FLDS opcode to add the **/event/deposit/release** event details to the **/purchased_deposit** object.

Validating a Deposit Release

Use the PCM_OP_DEPOSIT_POL_VALID_RELEASE_DEPOSIT opcode to validate and implement any customizations while releasing the deposit. This policy opcode is called by the PCM_OP_DEPOSIT_RELEASE_DEPOSIT opcode.

Transferring Deposits

Use the PCM_OP_DEPOSIT_TRANSFER_DEPOSIT opcode to transfer deposits from one account to another. The interest is calculated and updated before transferring and is also transferred to the target account.

The PCM_OP_DEPOSIT_TRANSFER_DEPOSIT opcode does the following:

- Validates the input flist.
- Calls the PCM_OP_DEPOSIT_POL_VALID_TRANSFER_DEPOSIT opcode to validate the input parameters.
- Reads the **/purchased_deposit** object using the PCM_OP_READ_OBJ opcode.
- Prepares and calls the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate the interest on the source account.
- Prepares the input flist to create the **/purchased_deposit** object on the target account or service by copying the relevant fields from the **/purchase_deposit** object.
- Calls the PCM_OP_CREATE_OBJ opcode to create the **/purchased_deposit** object on the target account.
- Calls the PCM_OP_ACT_USAGE opcode to create the **/event/deposit/transfer** event on the source account.
- Calls the PCM_OP_ACT_USAGE opcode to create the **/event/deposit/receive** event on the target account.

Validating a Deposit Transfer

Use the PCM_OP_DEPOSIT_POL_VALID_TRANSFER_DEPOSIT opcode to validate and implement any customizations while transferring the deposit. This policy opcode is called by the PCM_OP_DEPOSIT_TRANSFER_DEPOSIT opcode.

Adding Interest to a Deposit

Use the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate the interest amount for the purchased_deposit, for a specific duration.

The PCM_OP_DEPOSIT_ADD_INTEREST opcode does the following:

- Validates the input.
- Calls the PCM_OP_DEPOSIT_POL_PRE_ADD_INTEREST opcode to modify the effective date of the interest calculation.
- Calculates the interest based on the effective date or the interest calculation offset unit. The offset unit can be daily, weekly, monthly, or yearly.

> **Note:**
>
> If the effective date is passed, the interest is calculated from the last interest calculation date until the effective date for the applicable days. Otherwise, the interest is calculated based on the interest offset unit and the offset value.

- Calls the PCM_OP_DEPOSIT_POL_POST_ADD_INTEREST opcode to modify the calculated interest amount.
- Calls the PCM_OP_ACT_USAGE opcode to create the **/event/billing/deposit/interest/credit** event and associate it with the **/item/deposit/interest/credit** item. Sets PIN_FLD_START_T and PIN_FLD_END_T as interest calculation start and end dates from the PIN_FLD_INTEREST_INFO array.
- Calls the PCM_OP_ACT_USAGE opcode to create the **/event/billing/deposit/interest/debit** event and the **/item/deposit/interest/debit** item to negate the credited interest amount.
- Calls the PCM_OP_WRITE_FLDS opcode to add the **/event/billing/deposit/interest/debit** event details to the PIN_FLD_EVENTS array.
- Calls the PCM_OP_BILL_ITEM_TRANSFER opcode to transfer the amount from the **/item/deposit/interest/credit** object to the **/item/deposit/interest/debit** object.

Customizing the Interest Calculation

You can customize the interest calculation using these policy opcodes:

- PCM_OP_DEPOSIT_POL_PRE_ADD_INTEREST: Use to modify the effective date of the interest calculation.
- PCM_OP_DEPOSIT_POL_POST_ADD_INTEREST: Use to modify the calculated interest amount.

Requesting Deposit Refunds

Use the PCM_OP_DEPOSIT_REFUND_REQUEST opcode to create a deposit refund request.

The PCM_OP_DEPOSIT_REFUND_REQUEST opcode does the following:

- Validates the input.

- Calls the PCM_OP_DEPOSIT_POL_VALID_REFUND_REQUEST opcode to validate the input parameters.
- Calls the PCM_OP_CREATE_OBJ opcode to create the **/deposit_refund_request** request.
- Calls the PCM_OP_CUST_SET_NOTE opcode to create **/note** object. Creates the **/event/customer/note/create** event in the output flist.
- Calls the PCM_OP_ACT_USAGE opcode and creates the **/event/deposit/refund/request** event.

Validating a Deposit Refund Request

Use the PCM_OP_DEPOSIT_POL_VALID_REFUND_REQUEST opcode to validate and implement any customizations while requesting a deposit refund. This policy opcode is called by the PCM_OP_DEPOSIT_REFUND_REQUEST opcode.

Getting Refund Requests

Use the PCM_OP_DEPOSIT_GET_REFUND_REQUEST opcode to get all the deposit refund requests. This opcode is called by Billing Care.

The PCM_OP_DEPOSIT_GET_REFUND_REQUEST opcode does the following:

- Validates the input.
- Gets the user ID and adds it to the search template if the account or service POID or status exists.
- Calls the PCM_OP_SEARCH opcode to get, prepare, and return the **/deposit_refund_request**.

Updating Refund Requests

Use the PCM_OP_DEPOSIT_UPDATE_REFUND_REQUEST opcode to update the deposit refund review request. CSRs review the **/deposit_refund_request** object and select either APPROVE_FULL, APPROVE_PARTIAL, or REJECT.

The PCM_OP_DEPOSIT_UPDATE_REFUND_REQUEST opcode does the following:

- Validates the input.
- Gets the **/deposit_specification** object using the **/purchased_deposit** object.
- Gets the **/deposit_spec_profile** object using the **/deposit_specification** object.
- If the complete balance is released or refunded, calls the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate and add the interest amount.
- Calls the PCM_OP_ACT_USAGE opcode to charge the **/event/billing/product/fee/purchase/deposit/refund** event and associate it with the **/item/deposit/refund/fee** item.
- Calls the PCM_OP_BILL_ITEM_TRANSER opcode to transfer the **/item/deposit/refund/fee** item to the **/item/payment** item.
- Calls the PCM_OP_CUST_SET_NOTE opcode to create the **/note** object.
- Calls the PCM_OP_ACT_USAGE opcode to create an **/event/deposit/refund/approved** event.

- Calls the PCM_OP_ACT_USAGE opcode to charge the **/event/billing/product/fee/purchase/deposit/refund** event and associate it with the **/item/deposit/refund/fee** item.
- Calls the PCM_OP_BILL_ITEM_TRANSER opcode to transfer the **/item/deposit/refund/fee** item to the **/item/payment** item and transfer the approved_amount from the **/item/payment** item to the **/item/refund** item.

Updating Received Deposits

The PCM_OP_DEPOSIT_UPDATE_DEPOSIT_RECEIVED opcode is called through the notification of the **/event/billing/item/transfer** event to update the received amount, balance amount, and map the **/event/billing/item/transfer** event.

The PCM_OP_DEPOSIT_UPDATE_DEPOSIT_RECEIVED opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_WRITE_FLDS opcode to update the **/purchased_deposit** object with the input flist.
- Prepares the input flist for the PCM_OP_ACT_USAGE opcode to trigger the notification event, **/event/deposit/collected**.
- Calls the PCM_OP_ACT_USAGE opcode to update the deposit amount received.
- Calls the PCM_OP_BILL_SET_LIMIT_AND_CR opcode to decrease the credit limit by the deposit amount.

Triggering Deposits

The PCM_OP_DEPOSIT_TRIGGER_DEPOSIT opcode is called to create the **/purchased_deposit** object. This is called when any charge offer with the **/event/billing/product/fee/purchase/deposit** event type is created as part of purchasing a deal.

The PCM_OP_DEPOSIT_TRIGGER_DEPOSIT opcode does the following:

- Prepares the input flist.
- Searches for the deposit specification that is configured with the purchased product.
- Calls the PCM_OP_PURCHASE_DEPOSIT opcode to purchase the product that is associated with the deposit specification.

Retrieving Transaction Details of a Deposit

Use the PCM_OP_DEPOSIT_GET_TRANSACTIONS opcode to get the transaction details of a purchased deposit. It returns the complete details of the events. If this opcode is called for purchased deposit which is having a different parent, it fetches the event details of the purchased deposit in the hierarchy.

The PCM_OP_DEPOSIT_GET_TRANSACTIONS opcode does the following:

- Validates the mandatory parameters in the input flist.
- Gets the event details and corresponding notes for a given purchase deposit. If the input POID of purchase deposit is not same as the POID of parent or purchased_deposit, then it fetches the event and corresponding notes of the purchased deposit in the hierarchy.
- Calls the PCM_OP_SEARCH opcode for /event and corresponding notes object and prepares the output flist.

11

Device Management Opcodes

Learn about the tasks you can perform using the Oracle Communications Billing and Revenue Management (BRM) Device Manager APIs.

For more information, see "Managing Devices with BRM" in *BRM Developer's Guide*.

Topics in this document:

- Opcodes Described in This Chapter
- About the Device Management Opcodes
- Creating /device Objects
- Changing Device Attributes and States
- Associating /service and /device Objects
- Deleting /device Objects

Opcodes Described in This Chapter

Table 11-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 11-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_APN_POL_DEVICE_SET_STATE	Changing the State of a /device Object
PCM_OP_DEVICE_ASSOCIATE	Associating /service and /device Objects Changing the State of a /device Object Deleting /device Objects
PCM_OP_DEVICE_CREATE	Creating /device Objects
PCM_OP_DEVICE_DELETE	Deleting /device Objects
PCM_OP_DEVICE_POL_ASSOCIATE	Associating /service and /device Objects
PCM_OP_DEVICE_POL_CREATE	Creating /device Objects
PCM_OP_DEVICE_POL_DELETE	Deleting /device Objects
PCM_OP_DEVICE_POL_SET_ATTR	Changing the Attributes of /device Objects
PCM_OP_DEVICE_POL_SET_STATE	Changing the State of a /device Object

Table 11-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEVICE_SET_ATTR	Changing the Attributes of /device Objects
PCM_OP_DEVICE_SET_STATE	Changing the State of a /device Object Creating /device Objects
PCM_OP_DEVICE_UPDATE	Changing Device Attributes and States
PCM_OP_IP_POL_DEVICE_ASSOCIATE	Associating /service and /device Objects
PCM_OP_IP_POL_DEVICE_CREATE	Creating /device Objects
PCM_OP_IP_POL_DEVICE_DELETE	Deleting /device Objects
PCM_OP_IP_POL_DEVICE_SET_ATTR	Changing the Attributes of /device Objects
PCM_OP_NUM_POL_DEVICE_DELETE	Deleting /device Objects

About the Device Management Opcodes

You use the following opcodes to manage devices in BRM:

- To create devices, use PCM_OP_DEVICE_CREATE and PCM_OP_DEVICE_POL_CREATE. See "Creating /device Objects".
- To change any combination of device attributes and states, use PCM_OP_DEVICE_UPDATE. This wrapper opcode calls these opcodes to make the separate changes in a single transaction:
  - PCM_OP_DEVICE_SET_ATTR and PCM_OP_DEVICE_POL_SET_ATTR change existing device attribute values. See "Changing the Attributes of /device Objects".
  - PCM_OP_DEVICE_SET_STATE and PCM_OP_DEVICE_POL_SET_STATE change device states.

  You can also call these opcodes individually.
- To associate or disassociate services with devices, use PCM_OP_DEVICE_ASSOCIATE and PCM_OP_DEVICE_POL_ASSOCIATE. See "Associating /service and /device Objects".
- To delete devices, use PCM_OP_DEVICE_DELETE and PCM_OP_DEVICE_POL_DELETE. See "Deleting /device Objects".

Creating /device Objects

Use PCM_OP_DEVICE_CREATE to create **/device** objects in the BRM database.

PCM_OP_DEVICE_CREATE checks the device type on the input flist (PIN_FLD_OBJ_TYPE) to determine whether to call further device FM opcodes. For example, if the device type is ***/device/ip***, this opcode calls PCM_OP_DEVICE_POL_CREATE, which in turn calls PCM_OP_IP_POL_DEVICE_CREATE.

PCM_OP_DEVICE_CREATE calls PCM_OP_DEVICE_POL_CREATE before it creates any ***/device*** objects. This opcode is an empty hook, but you can customize it to validate IDs, check that all mandatory attributes are specified, or perform other custom tasks.

For example, if devices of a particular type require a device ID with certain characteristics, you can validate the ID supplied by the input flist. Similarly, you can use the opcode to ensure that all mandatory attributes of a particular device type are included in the new object.

PCM_OP_DEVICE_POL_CREATE probes the device type (PIN_FLD_OBJ_TYPE field) and may call other device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_IP_POL_DEVICE_CREATE to perform the validation checks it contains.

To create devices, your custom device-creation application must call PCM_OP_DEVICE_CREATE directly and pass in the following data:

- A type-only POID that specifies the device type
- The target database schema
- The name of the program calling the opcode
- The physical ID of the device
- The initial state to which the device should be set

> **Note:**
>
> Depending on the device type, additional inputs may be required for other device attributes.

PCM_OP_DEVICE_CREATE performs these tasks:

1. Calls PCM_OP_DEVICE_POL_CREATE to perform any custom validation.
2. Checks **/config/device_permit_map** to validate any device-to-service associations.
3. Creates a **/device** object of the type specified in the input flist. This object includes all mandatory attributes for the device type.
4. Calls PCM_OP_DEVICE_SET_STATE to set the initial device state.
5. If events are being recorded, generates an **/event/device/create** object.

PCM_OP_DEVICE_CREATE stops processing under these circumstances:

- When the validation of the device-to-service association fails.
- When PCM_OP_DEVICE_POL_CREATE fails during the validation process.
- When PCM_OP_DEVICE_SET_STATE fails while setting the object's initial device state.

If PCM_OP_DEVICE_CREATE is not successful, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back and no **/device** object is created.

Changing Device Attributes and States

Use PCM_OP_DEVICE_UPDATE to change any combination of device attribute values and states with a single transaction.

This opcode is called by BRM GUI applications and BRM FMs that modify device characteristics.

PCM_OP_DEVICE_UPDATE calls the following opcodes to perform validation checks before committing any changes:

- PCM_OP_DEVICE_POL_SET_ATTR
- PCM_OP_DEVICE_POL_SET_STATE

These individual opcodes in turn, call their respective policy opcodes to perform any validation checks that you have added as explained in the following sections.

Changing the Attributes of /device Objects

Use PCM_OP_DEVICE_SET_ATTR to change device attribute values, such as the device manufacturer and model number.

Each type of device can have different attributes. For example, a **/device/sim** object has different attributes than a **/device/num** object. Attributes common to all devices, such as the text description stored in the PIN_FLD_DESCR field, are contained in the base **/device** object. Attributes specific to a particular type of device are stored in the **/device** subclass.

PCM_OP_DEVICE_SET_ATTR calls PCM_OP_DEVICE_POL_SET_ATTR before it sets attributes. For example, you can write code to validate that the device ID in the input flist conforms to the pattern for a particular device type. PCM_OP_DEVICE_POL_SET_ATTR is an empty hook, but you can customize it to validate device IDs or perform other validation tasks.

PCM_OP_DEVICE_POL_SET_ATTR probes the device type (PIN_FLD_OBJ_TYPE field) and may call other device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_IP_POL_DEVICE_SET_ATTR to perform the validation checks it contains.

To change device attributes, your custom device-creation application must call PCM_OP_DEVICE_SET_ATTR directly and pass in the following data:

- The POID of the **/device** object.
- The name of the program calling the opcode.
- The fields to change, along with their new values. If the opcode is changing the attributes of a **/device** subclass, it can include fields introduced in the subclass.

  You cannot use PCM_OP_DEVICE_SET_ATTR alone to change the device state or service association. If the input flist includes these fields, they are ignored.

PCM_OP_DEVICE_SET_ATTR performs these operations:

1. Calls PCM_OP_DEVICE_POL_SET_ATTR to perform any custom validation.
2. Updates the **/device** object to include the new attribute values.
3. If events are being recorded, generates an **/event/device/attribute** object.

PCM_OP_DEVICE_SET_ATTR stops processing under these circumstances:

- When PCM_OP_DEVICE_POL_SET_ATTR fails.
- When an attribute to be changed doesn't exist in the **/device** object.

PCM_OP_DEVICE_SET_ATTR checks the device type on the input flist (PIN_FLD_OBJ_TYPE) to determine whether to call further device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_DEVICE_POL_SET_ATTR, which in turn calls PCM_OP_IP_POL_DEVICE_SET_ATTR.

Changing the State of a /device Object

Use PCM_OP_DEVICE_SET_STATE to change the state of a **/device** object.

This opcode checks the validity of each state change by using the **/config/device_state** object for this device type. The **/config/device_state** object includes an array that lists valid transitions from each device state. You create these rules in a configuration file and load them into **/config/device_state** objects by running the **load_pin_device_state** utility.

PCM_OP_DEVICE_SET_STATE checks the device type on the input flist (PIN_FLD_OBJ_TYPE) to determine whether to call further device FM opcodes. For example, if the device type is **/device/apn**, this opcode calls PCM_OP_DEVICE_POL_SET_STATE, which in turn calls PCM_OP_APN_POL_DEVICE_SET_STATE.

PCM_OP_DEVICE_SET_STATE calls policy opcodes at two points. The first takes place during the state change itself and the other just after it. You specify which policy opcodes to call by using the **pin_device_state** file. By default, it calls PCM_OP_DEVICE_POL_SET_STATE.

> **Note:**
>
> PCM_OP_DEVICE_SET_STATE uses the event notification feature. Before using this opcode, you must configure event notification for device management.

To set device states, your custom device-creation application must call PCM_OP_DEVICE_SET_STATE directly and pass in the following data:

- The POID of the **/device** object.
- The name of the program calling the opcode.
- The old state ID.
- The new state ID.

PCM_OP_DEVICE_SET_STATE performs these operations:

1. Determines whether the state change is valid by checking the **config/device_state** object for this device type.
2. Calls the opcode specified in **/config/device_state** for the in-transition event.
3. Changes the **/device** object's PIN_FLD_DEV_STATE_ID field to the new state.
4. If events are being recorded, generates an **/event/device/state** object.
5. Calls the opcode specified in **/config/device_state** for the post-transition event.

PCM_OP_DEVICE_SET_STATE stops processing under these circumstances:

- When the device state change is invalid based on the **/config/device_state** object for this device type.
- When a **/config/device_state** object cannot be found for this device type.
- When the input flist attempts to change the device state from RAW to any type other than INIT, or attempts to change the device state from END to any other state.
- When an error occurs during the event notification process or when a policy opcode is called.

> **Note:**
>
> The second policy call occurs after the state change has been recorded. If this policy fails, the state change is *not* rolled back.

PCM_OP_DEVICE_POL_SET_STATE allows customization during device state changes. For example, you might want to customize the process for assigning a SIM card to a customer. During this process, the state is changed from Inventory to Assigned. During the first policy call by PCM_OP_DEVICE_SET_STATE, the policy opcode could check the customer's handset to ensure compatibility with the SIM card. If the two devices are compatible, the state change takes place. In the second policy call, after the state change transaction is complete, the policy opcode could provision the SIM card by calling PCM_OP_DEVICE_ASSOCIATE.

PCM_OP_DEVICE_POL_SET_STATE probes the device type (PIN_FLD_OBJ_TYPE field) and may call other device FM opcodes. For example, if the device type is **/device/apn**, this opcode calls PCM_OP_APN_POL_DEVICE_SET_STATE to perform the validation checks it contains.

PCM_OP_DEVICE_POL_SET_STATE is an empty hook.

Associating /service and /device Objects

Use PCM_OP_DEVICE_ASSOCIATE to associate or disassociate services with devices.

When an account is provisioned with a device, this opcode associates the **/device** object with one or more **/service** objects. (One device can support multiple services. Conversely, one service may require multiple devices.) To facilitate searching, the account associated with the services is also associated with the device.

> **Note:**
>
> The services that can be associated with a particular device type are determined by the **/config/device_permit_map** object. You enter the device-to-service mapping information in a device-specific configuration file and load it into the database by running the **load_pin_device_permit_map** utility.

PCM_OP_DEVICE_ASSOCIATE calls PCM_OP_DEVICE_POL_ASSOCIATE before it associates any objects.

PCM_OP_DEVICE_POL_ASSOCIATE enables customized validation during device-to-service association and disassociation. For example, you could limit the number of associations for particular device types or trigger a state change after certain associations or disassociations.

PCM_OP_DEVICE_POL_ASSOCIATE probes the device type (PIN_FLD_OBJ_TYPE field) and may call other device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_IP_POL_DEVICE_ASSOCIATE to perform the validation checks it contains.

PCM_OP_DEVICE_POL_ASSOCIATE is an empty hook.

To associate services with devices, your custom device-creation application must call PCM_OP_DEVICE_ASSOCIATE directly and pass in the following data:

- The POID of the **/device** object.

- The name of the program calling the opcode.
- A flag to indicate association or disassociation.
- An array that includes the POIDs of the services and the account to be associated or disassociated.

PCM_OP_DEVICE_ASSOCIATE performs these operations:

1. Determines whether to associate or disassociate the objects by checking the PIN_FLD_FLAGS field.
   - When the last bit is **1**, the opcode *associates*.
   - When the last bit is **0**, the opcode *disassociates*.
2. Determines whether the device is already associated with the specified service by checking the **/device** object.
3. When associating objects, validates the device-to-service association by checking the **/config/device_permit_map** object.
4. Calls PCM_OP_DEVICE_POL_ASSOCIATE to perform any custom validation.
5. When associating objects, writes new fields for the services and account into the **/device** object's PIN_FLD_SERVICES array.
6. When disassociating objects, deletes the appropriate service and account fields from the **/device** object.
7. If events are being recorded, generates an **/event/device/associate** object.

PCM_OP_DEVICE_ASSOCIATE stops processing under these circumstances:

- For associations, when a service listed in the input flist already exists in the **/device** object.
- When the validation of the device-to-service association fails.

If the opcode stops processing, it logs an error in the CM **pinlog** file, indicating the reason for the failure. The transaction is rolled back and no association or disassociation takes place.

Deleting /device Objects

Use PCM_OP_DEVICE_DELETE to delete **/device** objects.

PCM_OP_DEVICE_DELETE checks the device type on the input flist (PIN_FLD_OBJ_TYPE) to determine whether to call further device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_DEVICE_POL_DELETE, which in turn calls PCM_OP_IP_POL_DEVICE_DELETE.

PCM_OP_DEVICE_DELETE calls PCM_OP_DEVICE_POL_DELETE before it deletes any device objects. By default, this opcode checks whether the device is associated with any services, and if it is, stops the transaction. You can customize this opcode to bypass this check or to perform other validation steps. For example, you can disable the service association check that is performed by default. You can also include a call to PCM_OP_DEVICE_ASSOCIATE to automatically disassociate services before device deletion.

PCM_OP_DEVICE_POL_DELETE checks whether a device is associated with any services, and if it is, stops the transaction. By default, PCM_OP_DEVICE_POL_DELETE stops processing under these circumstances:

- When the device is still associated with a service.
- When there is an error in the syntax of the input flist.

PCM_OP_DEVICE_POL_DELETE probes the device type (PIN_FLD_OBJ_TYPE field) and may call other device FM opcodes. For example, if the device type is **/device/ip**, this opcode calls PCM_OP_IP_POL_DEVICE_DELETE to perform the validation checks it contains.

PCM_OP_DEVICE_POL_DELETE calls a different opcode to customize device deletion. For example, if PIN_FLD_OBJ_TYPE is **/device/num**, this opcode calls PCM_OP_NUM_POL_DEVICE_DELETE to perform the validation checks it contains.

To delete devices, your custom device-creation application must call PCM_OP_DEVICE_DELETE directly and pass in the following data:

- The POID of the **/device** object.
- The name of the program calling the opcode.

PCM_OP_DEVICE_DELETE performs these operations:

1. Calls PCM_OP_DEVICE_POL_DELETE for validation.
2. Deletes the object.
3. If events are being recorded, generates an **/event/device/delete** object.

The opcode stops processing if PCM_OP_DEVICE_POL_DELETE fails.

12

General Ledger Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) general ledger opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- How G/L IDs Affect Storage and Reporting of G/L Data
- How BRM Records the Difference between Rounded and Unrounded Items
- Customizing G/L Data Stored in /journal Objects
- Customizing G/L Reports for Export
- Assigning G/L IDs to Prerated Events
- How BRM Exports G/L Reports
- How BRM Stores General Ledger Reports

Opcodes Described in This Chapter

Table 12-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 12-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_SPEC_GLID	Assigning G/L IDs to Prerated Events
PCM_OP_GL_LEDGER_REPORT	How BRM Exports G/L Reports How BRM Stores General Ledger Reports
PCM_OP_GL_POL_EXPORT_GL	Customizing G/L Reports for Export Customizing G/L Export Report Contents How BRM Exports G/L Reports
PCM_OP_GL_POL_PRE_UPDATE_JOURNAL	Customizing G/L Data Stored in /journal Objects

How G/L IDs Affect Storage and Reporting of G/L Data

The G/L ID assigned to a balance impact determines whether the balance impact is stored in a **/journal** object and whether it is included in a G/L report as shown in Table 12-2:

Table 12-2 G/L IDs

G/L ID	Description	Stored in Journal Object?	Included in G/L Report?
0	Default G/L ID. If you forget to assign a G/L ID to a balance impact type, BRM automatically assigns G/L ID 0 to it. You should reassign such balance impact types to an appropriate G/L ID.	Yes	No
1 through **99**	Assigned to balance impact types that *are not* included in G/L reports.	No	No
100 and above	Assigned to balance impact types that *are* included in G/L reports.	Yes	Yes

How BRM Records the Difference between Rounded and Unrounded Items

The difference between rounded and unrounded item totals is typically a very small amount referred to as the delta. This delta is recorded and included in G/L reports so that they can be accurately reconciled. The delta rounding difference is recorded in the following ways:

- When billing is run and pending items are rounded, the delta rounding difference is recorded in the **/item** object.
- A corresponding **/journal** entry object is created for each pending item. The delta rounding difference is stored as a credit or debit in the **/journal** object to balance the G/L:
  - If the rounding difference is negative, the balance needs to be credited, so the amount is stored in the PIN_FLD_CR_AR_NET_AMT field.
  - If the rounding difference is positive, the balance needs to be debited, so the amount is stored in the PIN_FLD_DB_AR_NET_AMT field.
- When a G/L report is generated, the journal entry containing the delta rounding difference is included in the report.

The delta rounding difference is calculated as the difference between a rounded pending item total and the sum of the rounded journal entries associated with the pending item:

```
    round(unrounded pending items)
 -  sum(round(journal entries)
-----------------------------------------
 =  rounding difference
```

For example, if a rounded pending item total is 11.72 and the sum of rounded journal entries is 11.73, the delta rounding difference is 11.72 - 11.73, or -.01.

Customizing G/L Data Stored in /journal Objects

During event processing, BRM retrieves balance impacts from each event and adds it to the revenue summary in the appropriate **/journal** object.

You can customize the data before it is written into **/journal** objects by using PCM_OP_GL_POL_PRE_UPDATE_JOURNAL.

PCM_OP_GL_POL_PRE_UPDATE_JOURNAL receives in the input flist the entries that are recorded in the **/journal** object. By default, the opcode does not perform any processing and returns only the entries passed in the input flist.

You can customize the opcode to add, modify, or remove one or more of the entries that will be recorded in the **/journal** object.

The opcode returns the following in the output flist:

- PIN_FLD_POID field set to the **/event** object POID.
- (Optional) PIN_FLD_JOURNAL_INFO array containing the entries that are recorded in the **/journal** object.

> **Note:**
>
> If the PIN_FLD_JOURNAL_INFO array is passed in the output flist, its PIN_FLD_POID field must be set either to **-1** (when creating a new **/journal** object) or to a valid **/journal** POID (when updating an existing **/journal** object).

By default, BRM does not call PCM_OP_GL_POL_PRE_UPDATE_JOURNAL before writing data into **/journal** objects. You configure BRM to call the opcode by modifying the **billing** instance of the **/config/business_params** object with the **pin_bus_params** utility.

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

To enable custom updates to G/L data:

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

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

3. In the file, change **disabled** to **enabled**:

   ```
   <CustomJournalUpdate>enabled</CustomJournalUpdate>
   ```

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

   ```
   pin_bus_params bus_params_billing.xml
   ```

6. Stop and restart the CM.

Customizing G/L Reports for Export

You can customize the data in your exported G/L reports by configuring PCM_OP_GL_POL_EXPORT_GL. The **pin_ledger_report** utility calls this opcode in **-export** mode after it generates a G/L report but before it exports the G/L report data to an XML file.

For example, you can map the G/L revenue account name in the BRM system to a different account name based on your company's criteria. To do this, you customize PCM_OP_GL_POL_EXPORT_GL to do the mapping and return the modified PIN_FLD_GL_ACCTS array from the input flist.

When the **pin_ledger_report** utility runs in **-export** mode, it creates a new ledger report object with the custom data in the PIN_FLD_GL_ACCTS array. This customized G/L report is used for further processing and is exported to the G/L report XML files.

The PIN_FLD_RESULT value in the output flist determines whether to create a customized report from the original report used for export:

- **0**: Use the original G/L data to generate the export report. This is the default.
- **1**: Use the G/L data in the output flist to generate a custom export report.

> **Note:**
>
> The values returned by PCM_OP_GL_POL_EXPORT_GL must be in summary format even when the input flist data is in detailed format.

Customizing G/L Export Report Contents

Use PCM_OP_GL_POL_EXPORT_GL to customize the contents of your exported G/L reports. For example, the BRM G/L account names could be mapped to the third-party account names or the BRM balance element IDs could be mapped to the third-party balance element IDs.

Mapping BRM G/L Account Names to Third-Party Account Names

Use PCM_OP_GL_POL_EXPORT_GL to convert BRM G/L account names to the G/L account names in your external G/L system before XML reports are exported. This makes importing them into the external system easier. The following is an example of the code used to map BRM G/L account names to the third-party account names after PCM_OP_GL_POL_EXPORT_GL has been called.

```
#*********************************************************
#Section to customize report
#*********************************************************
PCM_OP(ctxp, PCM_OP_READ_OBJ, 0, in_flistp, &r_flistp, ebufp);
PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG,"Flistp after read obj in Custom report",
r_flistp);
while ((tmp_flistp = PIN_FLIST_ELEM_TAKE_NEXT(r_flistp, PIN_FLD_GL_ACCTS,&rec_id, 1,
&cookie, ebufp)) != (pin_flist_t *)NULL)
{
         char *segment = NULL;
         PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "GL accts from input flist", tmp_flistp);
         tmp1_flistp = PIN_FLIST_ELEM_ADD(out_flistp, PIN_FLD_GL_ACCTS, ctr, ebufp);
         PIN_FLIST_CONCAT(tmp1_flistp, tmp_flistp, ebufp);
         ctr++;
```

```
            vp = PIN_FLIST_FLD_GET(r_flistp, PIN_FLD_GL_SEGMENT, 0, ebufp);
            if (vp)
           {
               segment = (char *)vp;
            }
            modify_acct_name(tmp1_flistp, PIN_FLD_AR_GROSS_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_OFF_GROSS_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_AR_NET_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_OFF_NET_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_AR_DISC_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_OFF_DISC_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_AR_TAX_GL_ACCT, segment, ebufp);
            modify_acct_name(tmp1_flistp, PIN_FLD_OFF_TAX_GL_ACCT, segment, ebufp);
            PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "GL accts from output flist ",
tmp1_flistp);
            PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "Intermediate output flist ",
out_flistp);
}
#*********************************************************
#Set PIN_FLD_RESULT to indicate customization
#*********************************************************/
result = 1;
PIN_FLIST_FLD_SET(out_flistp, PIN_FLD_RESULT, (void *)&result, ebufp);
```

Mapping BRM Balance Element IDs to Third-Party Balance Element IDs

If the mapping capabilities provided with the BRM **pin_glid** mapping file are not sufficient, you can use PCM_OP_GL_POL_EXPORT_GL to convert BRM balance element IDs to the balance element IDs in your external G/L system before XML reports are exported. This makes importing them into the external system easier. The following is an example of the code used to map BRM balance element IDs to the third-party balance element IDs after PCM_OP_GL_POL_EXPORT_GL has been called.

```
#*********************************************************
#Section to customize report
#*****************************************************************
PCM_OP(ctxp, PCM_OP_READ_OBJ, 0, in_flistp, &r_flistp, ebufp);
PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "Flistp after read obj in Custom report",
r_flistp);
while ((tmp_flistp =PIN_FLIST_ELEM_TAKE_NEXT(r_flistp, PIN_FLD_GL_ACCTS,&rec_id, 1,
&cookie, ebufp)) != (pin_flist_t *)NULL)
{
       PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "GL accts from input flist ", tmp_flistp);
       tmp1_flistp = PIN_FLIST_ELEM_ADD(out_flistp, PIN_FLD_GL_ACCTS, ctr, ebufp);
       PIN_FLIST_CONCAT(tmp1_flistp, tmp_flistp, ebufp);
       ctr++;
       vp = PIN_FLIST_FLD_GET(r_flistp, PIN_FLD_RESOURCE_ID, 0, ebufp);
       if (vp)
          {
          resource_id = (int32 *)vp;
          if (resource_id == 840)
              {
              new_resource_id = 1840;
              }
          }
PIN_FLIST_FLD_SET(out_flistp, PIN_FLD_RESOURCE_ID, (void *)&new_resource_id, ebufp);
PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "GL accts from output flist ", tmp1_flistp);
PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, "Intermediate output flist ", out_flistp);
}
#*********************************************************
#Set PIN_FLD_RESULT to indicate customization
```

```
#*********************************************************
result = 1;
PIN_FLIST_FLD_SET(out_flistp, PIN_FLD_RESULT, (void *)&result, ebufp);
```

Customizing the G/L Report XML File Names

To customize the XML file names for your G/L reports, use the **FileNamePrefix** tag in the **pin_config_export_gl.xml** configuration file. For example, if your company tracks G/L data separately for multiple divisions of the same company, you can prefix the G/L data for each division with a unique identifier.

Assigning G/L IDs to Prerated Events

Use PCM_OP_ACT_POL_SPEC_GLID to assign G/L IDs to prerated events or partially rated events. By default, this opcode retrieves an event's G/L ID from the **/config/map_glid** object. You can customize it to perform searches on other objects and look at other fields.

For example, you can categorize adjustments by customizing the opcode to assign G/L IDs based on the adjustment type.

All customization is done using **pin_glid** and COA files.

How BRM Exports G/L Reports

The **pin_ledger_report** utility performs the following tasks when it runs:

1. Retrieves G/L report configuration information from the **/config/export_gl** object to determine which reports to generate and which G/L segments to report.
2. Creates a new **/process_audit/export_gl** object for this run and sets the status of the **/process_audit/export_gl** object to IN_PROGRESS.
3. Reads the previous **/process_audit/export_gl** object and the **ReportInitialStartDate** value in the **/config/export_gl** object to determine if any G/L reports for previous periods have not been created. If so, creates them.
4. Calls PCM_OP_GL_LEDGER_REPORT to create the **/ledger_report** object for the current period.
5. Using the POID of the generated **/ledger_report** object, calls PCM_OP_GL_POL_EXPORT_GL and then does one of the following, based on the PIN_FLD_RESULT value:
   - **0**: No customization; generates and exports the original report.
   - **1**: Customization. If the G/L data was updated, creates a new **/ledger_report** object in database and exports the new report.

   > **Note:**
   >
   > If the report contains cumulative revenue, generates an incremental report by using the G/L report for the previous period, if one exists, and updates the **/ledger_report** object.

6. If specified in the export configuration file, filters noncurrency balance elements from the generated **/ledger_report** object to be included in or excluded from the XML output file.

7. Generates the XML output file.
8. Updates the **/process_audit/export_gl** object with information about this run.
9. Posts reports in the database.
10. Sets the status of the **/process_audit/export_gl** object to RUN_COMPLETED.

> **Note:**
>
> If the **pin_ledger_report** utility encounters errors during processing, it exits without completing the report processing. (One or more prior reports and one or more G/L export reports *may* have been successfully generated.)

How BRM Stores General Ledger Reports

The **pin_ledger_report** utility uses PCM_OP_GL_LEDGER_REPORT to store general ledger reports in the BRM database. This opcode calls PCM_OP_CREATE_OBJ to create a **/ledger_report** object.

When the PCM_OPFLG_READ_RESULT flag is set, the opcode returns the entire contents of the **/ledger_report** object.

13

Group Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) group opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Managing Sharing Groups
- Managing Ordered Balance Groups
- Adding and Removing Members Automatically to Sharing Groups
- Displaying Balance Monitor Information in Client Applications
- Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications
- Managing Balance Monitors
- Managing Balance Groups
- Converting an Account Hierarchy to Wholesale Billing
- Rolling Up Non-Currency Balance from Members to the Owner of a Wholesale Sharing Group

Opcodes Described in This Chapter

Table 18-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 13-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BAL_APPLY_MONITOR_IMPACTS	Updating Monitor Balances and Sending Credit Limit/ Threshold Breach Notifications
PCM_OP_BAL_CHANGE_VALIDITY	Modifying the Sub-balance Validity Period
PCM_OP_BAL_GET_ACCT_MONITORS	Retrieving the Balance Monitors Owned by an Account or Service
PCM_OP_BAL_GET_MONITOR_BAL	Retrieving the Balances for a Monitor Group
PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC	Customizing the Default Balance Group of a Bill Unit
PCM_OP_BILL_GROUP_REMOVE_PAYING_MEMBERS	Removing Members from Sharing Groups

Table 13-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_SET_LIMIT_AND_CR	Creating Balance Groups Modifying Sub-balances
PCM_OP_CUST_COMMIT_CUSTOMER	Creating Balance Groups
PCM_OP_CUST_CONVERT_WHOLESALE_HIERARCHY	Converting an Account Hierarchy to Wholesale Billing
PCM_OP_CUST_CREATE_BAL_GRP	Creating Balance Groups
PCM_OP_CUST_DELETE_BAL_GRP	Deleting a Balance Group Deactivating Balance Monitors
PCM_OP_CUST_MODIFY_BAL_GRP	Creating Balance Groups
PCM_OP_CUST_MODIFY_CUSTOMER	Creating Balance Groups Moving a Balance Group from One Bill Unit to Another
PCM_OP_CUST_SET_BAL_GRP	Creating Balance Groups Creating and Updating Balance Monitors
PCM_OP_MONITOR_ACCOUNT_HIERARCHY	Adding and Removing Members Automatically to Sharing Groups Adding Members to Hierarchy-Type Sharing Groups Automatically
PCM_OP_MONITOR_BILLING_HIERARCHY	Adding and Removing Members Automatically to Sharing Groups Adding Members to Paying Responsibility-Type Sharing Groups Automatically
PCM_OP_MONITOR_HIERARCHY_CLEANUP	Adding and Removing Members Automatically to Sharing Groups Removing Members from Balance Monitor Groups
PCM_OP_MONITOR_PROCESS_BILLING_MONITORS	Adding Members to Payment Responsibility-Type Balance Monitors
PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS	Adding Members to Hierarchy-Type Balance Monitors
PCM_OP_MONITOR_SERVICE_HIERARCHY	Adding and Removing Members Automatically to Sharing Groups Updating Subscription-Type Monitors Automatically
PCM_OP_MONITOR_SETUP_MEMBERS	Adding and Removing Members Automatically to Sharing Groups Adding Members to Newly Created Sharing Groups Automatically
PCM_OP_MONITOR_UPDATE_MONITORS	Adding a Monitor Group to a Member's /ordered_balgrp Object

Table 13-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP	Managing Ordered Balance Groups Deleting a Sharing Group Creating an Ordered Balance Group Adding a Sharing Group to an Ordered Balance Group Adding a Monitor Group to a Member's /ordered_balgrp Object Deleting a Sharing Group from an Ordered Balance Group Customizing the Order in Which to Apply Sharing Groups Deleting an Ordered Balance Group
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY	Managing Ordered Balance Groups Creating and Modifying Multiple Ordered Balance Groups Simultaneously
PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS	Getting a List of Charges Available for Charge Sharing
PCM_OP_SUBSCRIPTION_POL_ORDERED_BALGRP_PRIORITY	Creating an Ordered Balance Group Adding a Sharing Group to an Ordered Balance Group Customizing the Order in Which to Apply Sharing Groups
PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS	Validating the Members of a Balance Monitor Group
PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING	Creating a Charge Sharing Group Adding Sponsored Charges to a Charge Sharing Group
PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE	Creating a Sharing Group
PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE	Deleting a Sharing Group
PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY	Modifying a Sharing Group
PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT	Changing the Owner of a Sharing Group Changing the Owner of a Balance Monitor
PCM_OP_SUBSCRIPTION_SHARING_GROUP_ROLLUP	Rolling Up Non-Currency Balance from Members to the Owner of a Wholesale Sharing Group

Managing Sharing Groups

To manage balance monitoring groups, charge sharing groups, discount sharing groups, profile sharing groups, and product sharing groups, use the following opcodes:

- PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE

  See "Creating a Sharing Group".
- PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY

  See "Modifying a Sharing Group".
- PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE

  See "Deleting a Sharing Group".
- PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT

  See "Changing the Owner of a Sharing Group".
- (Charge Sharing Groups Only) PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS

See "Getting a List of Charges Available for Charge Sharing".

Creating a Sharing Group

Use PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE to create a sharing group, including balance monitoring groups, charge sharing groups, discount sharing groups, profile sharing groups, or product sharing groups. The members of a sharing group can include individual services, all services of a specific type, or a group of accounts.

Observe the following guidelines when creating sharing groups:

- Products, discounts, and charges shared by the group owner must be valid, meaning that it has an active or inactive status and has not expired.
- If a member owns another sharing group, the owner of the sharing group you create cannot belong to the member's group.
- The name of a sharing group must be unique and cannot be used for another sharing group owned by the group owner.
- If the member is a service, the service can be either a service type or a specific service instance. When adding a type-only service, all instances of that service type become members. However, the subtypes of the service type do not become members.
- By default, the parent account and all group members must reside in the same database schema. To enable sharing groups to include members from other database schemas, enable the **CrossSchemaSharingGroup** business parameter. See "Enabling Group Members to Reside in Multiple Schemas" in *BRM Managing Customers*.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE returns the following:

- The POID of the sharing object that was created: **/group/sharing/products**, **/group/sharing/discounts**, **/group/sharing/charges**, **/group/sharing/monitor**, or **/group/sharing/profiles**.
- The POID of the event generated to record the creation: **/event/group/sharing/products/create**, **/event/group/sharing/discounts/create**, **/event/group/sharing/profiles/create**, **/event/group/sharing/monitor/create**, or **/event/group/sharing/charges/create**.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE fails in the following cases:

- The owner is also a member of a sharing group owed by one of the members.
- A product, charge, or discount in the input flist is not valid.
- The primary currency for any of the members is different from the owner's primary currency.

Creating a sharing group is the first step toward activating sharing. To apply shared discounts, charges, profiles, and products for the group's members, you must also create ordered balance groups for the members. See "Creating an Ordered Balance Group".

> **Note:**
>
> When creating a global charge sharing group, do not create or modify each member's ordered balance group.

When PIN_FLD_SHARING_ROLLUP is passed with the value 0 or not passed , a normal group is created. When passed with the value 1 it will create rollup sharing groups.

> **Note:**
>
> Rollup sharing group is applicable only to wholesale hierarchy.

> **Note:**
>
> You do not need to create ordered balance groups for members of global charge sharing groups.

Creating a Discount Sharing Group

To create a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does the following:

1. Creates a **/group/sharing/discounts** object and assigns the group owner.

   > **Note:**
   >
   > - The owner cannot be a member of its own group.
   > - If a member of the group owns a discount or charge sharing group, the owner cannot be a member of that group.

2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/discounts** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the discount sharing group. However, subclass instances of the service type do not become members.

   For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/discounts** object is created. The member account can join the group and purchase the service later. BRM begins discount sharing for the service as soon as the member buys the service and joins the discount sharing group.

3. Adds the **/discount** objects in the PIN_FLD_DISCOUNTS array to the DISCOUNTS array of the **/group/sharing/discounts** object.

   The discounts must be owned by the discount sharing group owner. They must also have validity dates and a valid status (active or inactive). If the PIN_FLD_DISCOUNTS array is empty, all of the owner's discounts are shared. In this case, the PIN_FLD_DISCOUNTS array in the **/group/sharing/discounts** object contains **NULL** in the PIN_FLD_DISCOUNT_OBJ and PIN_FLD_OFFERING_OBJ fields.

4. Generates an **/event/group/sharing/discounts/create** event to record the creation of the discount sharing group.

Creating a Charge Sharing Group

To create a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does the following:

1. Creates a **/group/sharing/charges** object, and assigns the owner to the group.

   > **Note:**
   >
   > The owner cannot be a member of its own group.

2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/charges** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the charge sharing group. However, subclass instances of the service type do not become members.

   For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/charges** object is created. The member account can join the group and purchase the service later. BRM begins charge sharing for the service as soon as the member buys the service and joins the charge sharing group. BRM handles future services for charge sharing in the same way that it does for discount sharing.

3. Validates any product specification attributes on the **/sponsorship** objects by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *Setting Up Pipeline Pricing*.

4. Adds **/sponsor** objects in the PIN_FLD_SPONSORS array to the SPONSORS array of the **/group/sharing/charges** object.

5. Generates an **/event/group/sharing/charges/create** event to record the creation of the charge sharing group.

Creating a Product Sharing Group

To create a product sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does the following:

1. Creates a **/group/sharing/products** object and assigns the owner to the group.

> **Note:**
>
> The owner cannot be a member of its own group. Also, an account can be the owner of only one product sharing group at a time.

2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/products** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the product sharing group. However, subclass instances of the service type do not become members. For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members but it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/products** object is created. The member account can join the group and purchase the service later. BRM begins sharing the product as soon as the member buys the service and joins the product sharing group. BRM handles future services for product sharing in the same way that it does for discount sharing.

3. Validates any product specification attributes on the **/sponsorship** objects by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *Setting Up Pipeline Pricing*

4. Adds **/sponsor** objects in the PIN_FLD_SPONSORS array to the SPONSORS array of the **/group/sharing/products** object.

5. Generates an **/event/group/sharing/products/create** event.

Creating a Balance Monitor Group

To create a balance monitor group (**/group/sharing/monitor** object), call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode and pass the following information in the opcode's input flist:

- Owner of the balance monitor:
  - When the owner is an account, the POID of the **/account** object
  - When the owner is a service, the POIDs of the **/account** and **/service** objects
- Sharing group type set to the type-only POID for **/group/sharing/monitor**
- Balance monitor (**/balance_group/monitor** object) to associate with the monitoring group
- List of members you want to add or delete

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does the following:

1. Verifies that balance monitoring is enabled.
2. Calls the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode to validate the members.

   See "Validating the Members of a Balance Monitor Group" for more information.

3. Creates a **/group/sharing/monitor** object and assigns the owner to the group.

   > **Note:**
   >
   > The owner cannot be a member of its own group. Also, an account can be the owner of only one balance monitor group at a time.

4. If the PIN_FLD_TYPE_STR input flist field is set to **PR_RTCE** and the PIN_FLD_ROLL_UP_CREDIT_LIMIT input flist field is set to **1**, does the following:
   a. Rolls up each member's credit limit to the owner's billing account.
   b. If the **ResetMemberCreditLimit** business parameter is enabled, sets each member's credit limit to NULL. See "Nullifying Credit Limits for PR_RTCE Child Members" in *BRM Managing Customers*.
5. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/products** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the balance monitor group. However, subclass instances of the service type do not become members. For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members but it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/products** object is created. The member account can join the group and purchase the service later. BRM begins sharing the product as soon as the member buys the service and joins the balance monitor group. BRM handles future services for balance monitoring in the same way that it does for discount sharing.
6. Validates any product specification attributes on the **/sponsorship** objects by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *Setting Up Pipeline Pricing*
7. Adds **/sponsor** objects in the PIN_FLD_SPONSORS array to the SPONSORS array of the **/group/sharing/monitor** object.
8. Generates an **/event/group/sharing/monitor/create** event.

Validating the Members of a Balance Monitor Group

To validate the members of a balance monitor, call the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode. This opcode validates members before they are added or modified based on the monitor type, and returns a list of valid members to the calling opcode.

By default, this opcode validates the hierarchy, paying responsibility, and service level monitor types. You can create any type of monitor and validate it by implementing validation rules in this opcode.

The default implementation of the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS opcode performs the following tasks:

1. Retrieves the monitor type.
2. For each member in the input flist, retrieves the account object.
3. Applies validation rules for the monitor type and returns a list of valid members for the group.

Modifying a Sharing Group

Use PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY to modify a product sharing group, discount sharing group, profile sharing group, or charge sharing group. You can modify sharing groups by:

- Adding members to an existing group.
- Adding discounts or charges to an existing group.
- Changing the sharing group to support rolling up member balances to the sharing group owner.

> **Note:**
>
> Do not change a rollup sharing group to a normal sharing group.

If successful, it returns the POID of the group that was modified and the POIDs of the events generated by PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY to record the group modification.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY fails in the following cases:

- If the owner is a member of its own group or a group owned by a group member.
- If a discount offer in the input flist is not valid.

Adding a member to a sharing group is the first step toward activating sharing. For the new member to benefit from shared products, profiles, discounts, and charges, you must also create an ordered balance group for the member. See "Creating an Ordered Balance Group".

> **Note:**
>
> You do not need to create ordered balance groups for members of global charge sharing groups.

Adding Members to a Discount or Charge Sharing Group

To add members to a discount or charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

> **Caution:**
>
> The element ID for each member in the input flist must be unique within the membership. If an element ID is already being used by an existing member in the group object you are modifying, the opcode overwrites the existing member. If you do not know the element IDs of each member in the object, you can prevent member loss by making sure the flist includes both existing and new members.

1. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the group object:
   - The **/group/sharing/charges** object for a charge sharing group.
   - The **/group/sharing/discounts** object for a discount sharing group.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

   > **Note:**
   >
   > - The owner cannot be a member of its own group.
   > - If a member of the group owns a discount or charge sharing group, the owner cannot be a member of that group.
   > - If the PIN_FLD_MEMBERS array is empty, this opcode calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE to delete all members from the group.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the discount sharing group. However, subclass instances of the service type do not become members.

   For example, if the input flist specifies the GSM service type, this opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

   If you specify a service type as a member, the member account does not need to own the service when the group object is modified. The member account can join the group and purchase the service later. BRM begins discount or charge sharing for the service as soon as the member buys the service and joins the discount or charge sharing group.
2. Generates an event to record member changes:
   - An **/event/group/sharing/discounts/modify** event for a discount sharing group.
   - An **/event/group/sharing/charges/modify** event for a charge sharing group.

Adding Discounts to a Discount Sharing Group

To add discounts for a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/discount** objects in the PIN_FLD_DISCOUNTS array to the **/group/sharing/discounts** object's discounts array.

- The discount instance must be owned by the group owner. They must also have validity dates and a valid status (active or inactive).
- If the PIN_FLD_DISCOUNTS array is empty, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE is called to delete all **/discount** objects from the **/group/sharing/discounts** object's discounts array.

2. Generates an **/event/group/sharing/discounts/modify** event to record the event.

Adding Sponsored Charges to a Charge Sharing Group

To add sponsored charges to a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Validates any product specification attributes on the **/sponsorship** object by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *BRM Setting Up Pipeline Pricing*.
2. Adds the **/sponsorship** objects in the PIN_FLD_SPONSORS array to the **/group/sharing/charges** object's SPONSORS array.

> **Note:**
>
> If the PIN_FLD_SPONSORS array is empty, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE is called to delete all **/sponsor** objects from the **/group/sharing/charges** object's SPONSORS array.

3. Generates an **/event/group/sharing/charges/modify** event to record the event.

Deleting a Sharing Group

Use PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE to delete a discount sharing group, charge sharing group, profile sharing group, or product sharing group. You can also use this opcode to delete shared discounts, sponsored charges, shared profiles, shared products, and group members.

- For a discount sharing group, this opcode deletes shared discounts, group members, or the sharing group itself.
- For a charge sharing group, this opcode deletes sponsored charges, group members, or the sharing group itself.
- For a profile sharing group, this opcode deletes shared profiles, group members, or the sharing group itself.
- For a product sharing group, this opcode deletes shared products, group members, or the sharing group itself.

Deleting a Discount Sharing Group

To delete a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. For each member in the group, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **/group/sharing/discounts** object from the member's **/ordered_balgrp** object.
2. Deletes the **/group/sharing/discounts** object.
3. Generates an **/event/group/sharing/discounts/delete** event to record the deletion.

Deleting a Charge Sharing Group

To delete a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. For each member in the group, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **/group/sharing/charges** object from the member's **/ordered_balgrp** object.
2. Deletes the **/group/sharing/charges** object.
3. Generates an **/event/group/sharing/charges/delete** event.

Deleting a Member from a Discount Sharing Group

To delete a member from a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

> **Caution:**
>
> The element ID for each member in the **/group/sharing/discounts** object is unique. If you use an incorrect element ID for the member you want to delete, this opcode deletes the wrong member.

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **/group/sharing/discounts** object from the member's **/ordered_balgrp** object.
2. Deletes the member from the **/group/sharing/discounts** members array.
3. Generates an **/event/group/sharing/discounts/delete** event.

Deleting a Member from a Charge Sharing Group

To delete a member from a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

> **Caution:**
>
> The element ID for each member in the **/group/sharing/charges** object is unique. If you use an incorrect element ID for the member you want to delete, this opcode deletes the wrong member.

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **/group/sharing/charges** object from the member's **/ordered_balgrp** object.
2. Deletes the member from the **/group/sharing/charges** members array.

3. Generates an **/event/group/sharing/charges/delete** event.

Deleting a Shared Discount from a Discount Sharing Group

To delete a shared discount from a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

> **Caution:**
>
> The OFFERING_OBJ uniquely identifies each shared discount in the **/group/sharing/discounts** object. If you use an incorrect OFFERING_OBJ for the discount you want to delete, this opcode deletes the wrong discount.

1. Deletes the **/discount** objects specified in the PIN_FLD_DISCOUNTS array from the DISCOUNTS array of the **/group/sharing/discounts** object.
2. Generates an **/event/group/sharing/discounts/delete** event to record the deletion.

Deleting a Sponsored Charge from a Charge Sharing Group

To delete a sponsored charge from a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

> **Caution:**
>
> The element ID for each shared charge in the **/group/sharing/charges** object is unique. If you use an incorrect element ID for the charge you want to delete, this opcode deletes the wrong charge.

1. Deletes the **/sponsor** objects specified in the PIN_FLD_SPONSORS array from the SPONSORS array of the **/group/sharing/charges** object.
2. Generates an **/event/group/sharing/charges/delete** event to record the deletion.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE returns the POID of the sharing group object that was modified and the POID of the event that was generated.

Changing the Owner of a Sharing Group

Use PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT to change the owner of a discount sharing group, charge sharing group, profile sharing group, or product sharing group.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT takes the following input:

- PIN_FLD_POID: The **/account** object of the current owner.
- PIN_FLD_GROUP_OBJ: The sharing group object that is being changed: **/group/sharing/discounts**, **/group/sharing/charges**, **/group/sharing/profiles**, or **/group/sharing/products**.
- PIN_FLD_ACCOUNT_OBJ: The **/account** object of the new owner.

> **Note:**
>
> If you are changing the ownership of a discount or charge sharing group from one service to another in the same account, the object passed in this field matches the one in the PIN_FLD_POID field.

- PIN_FLD_PARENT: The new owner, which is either an **/account** object or a **/service** object.
- PIN_FLD_BAL_GRP_OBJ: The **/balance_group** object used to track sharing activities for the new owner.
- The list of discounts, profiles, or products associated with the new owner.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT returns the POID of the sharing group that was modified and the event that was generated to record the owner change.

This opcode fails if the new owner is a member of the sharing group.

Changing the Owner of a Discount Sharing Group

To change the owner of a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies the following:
   - The input flist contains the **/balance_group** object and **/account** object for the new discount sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - The currency of the new owner is the same as the currency of the old owner.
   - There are no circular relationships. For example, the owner cannot be a member of its own group. Also, if a member of the group owns a sharing group, the owner cannot be a member of that group.
2. Deletes the current owner's discounts from the **/group/sharing/discounts** object.
3. Sets the **/group/sharing/discounts** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
4. Adds the new owner's shared discounts to the **/group/sharing/discounts** object.
5. Generates an **/event/group/sharing/discounts/modify** event to record the owner change.

Changing the Owner of a Charge Sharing Group

To change the owner of a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies the following:
   - The input flist contains the **/balance_group** object and **/account** object for the new charge sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - The currency of the new owner is the same as the currency of the old owner.

- There are no circular relationships. For example, the owner cannot be a member of its own group.

2. Sets the **/group/sharing/charges** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
3. Updates the sponsor flags in the old owner's and new owner's bill units (**/billinfo** objects).
4. Generates an **/event/group/sharing/charges/modify** event to record the owner change.

Changing the Owner of a Balance Monitor

To change the owner of a balance monitor (**/group/sharing/monitor** object), call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT opcode.

> **Note:**
>
> This opcode can be used to change the owner of a balance monitor only if you are performing the change manually. Automated Monitor Setup (AMS) does not support changing owners.

If successful, this opcode returns the POID of the balance monitor that was modified and the event that was generated to record the owner change.

This opcode fails if the new owner that is passed is already a member of the resource sharing group.

To change the owner of a balance monitor, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - AMS is not enabled. See "Enabling AMS in BRM" in *BRM Managing Customers*.
   - The input flist contains the **/balance_group** object and **/account** object for the new balance monitor owner.
   - The **/balance_group** object belongs to the new owning account or service.
2. Sets the **/group/sharing/monitor** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
3. Creates an **/event/group/sharing/monitor/modify** event to record the owner change.

Adding a Monitor Group to a Member's /ordered_balgrp Object

The **/ordered_balgrp** object stores the list of sharing groups to which an account or service belongs. This object controls the order in which credits and discounts are applied and tracks the balance impacts for the balance groups.

Monitor groups are added to the PIN_FLD_ORDERED_BALGROUPS array of the **/ordered_balgrp** object. Unlike the charge sharing and discount sharing groups, the order of the monitor groups does not affect the balances. Therefore, the monitor groups are added to the end of the list.

To add a monitor group to the **/ordered_balgrp** object of an account or service, call the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. This opcode performs the following tasks, depending on the value passed in the PIN_FLD_ACTION field in the input flist:

- Creates or deletes an **/ordered_balgrp** object.
- Modifies an **/ordered_balgrp** object by adding or deleting groups.
- Lists all sharing groups to which the account or service belongs.

This opcode performs the following tasks:

1. Verifies that the input flist has a **/group/sharing/monitor** object in the PIN_FLD_ORDERED_BALGROUPS array and that balance monitoring is enabled.

   > **Note:**
   >
   > - When the input flist does not contain a /group/sharing/monitor object, the opcode continues with the tasks for processing the /group/sharing/charges and /group/sharing/discounts objects.
   > - When balance monitoring is not enabled, the opcode returns an error.

2. Rearranges the PIN_FLD_ORDERED_BALGROUPS array to place the monitor groups at the end of the sequence.
3. Does one of the following tasks:
   - When the opcode is called to *create* the object, creates the **/ordered_balgrp** object for the specified account or service.
   - When the opcode is called to *modify* the object, modifies the **/ordered_balgrp** object by adding or deleting monitor group objects.
   - When the opcode is called to *delete* the object, deletes the **/ordered_balgrp** object.
4. Retrieves the current balance for the account or service **/balance_group**.
5. Calls PCM_OP_MONITOR_UPDATE_MONITORS to generate an **/event/billing/monitor/update** event (when an **/ordered_balgrp** object is created or modified) or an **/event/billing/monitor/delete** event (when an **/ordered_balgrp** object is deleted).
6. Updates the balance of all associated **/balance_group/monitor** objects.

Getting a List of Charges Available for Charge Sharing

To get a list of all charges available for charge sharing, use PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS.

When you create a charge sharing group, you select the charges, or chargeshares, that you want the owner to assume for the members. Chargeshares are stored in **/sponsorship** objects in the BRM database. By default, this opcode retrieves all the **/sponsorship** objects.

You can customize PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS to filter or return a subset of **/sponsorship** objects from the list of all available **/sponsorship** objects. For example, you can customize this opcode to get a list of the chargeshares that include a particular set of events.

PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS is not called by any opcode.

Managing Ordered Balance Groups

Use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to create, modify, or delete an ordered balance group (**/ordered_balgrp** object) for an account or service that is a member of a sharing group.

The ordered balance group contains links to the sharing groups that the member has joined, listed in order by rank. For discount sharing groups, charge sharing groups, and product sharing groups, the rank controls the order in which the group's balances are impacted by events. For profile sharing groups, the rank is not significant.

> **Note:**
>
> PCM_OP_SUBSCRIPTION_ORDERED_BALGRP creates or modifies one ordered balance group at a time. To create or modify more than one ordered balance group at time, use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

An ordered balance group object is created when an account or service joins a sharing group. You can modify an ordered balance group as follows:

- Add a sharing group
- Delete a sharing group

The **/ordered_balgrp** object stores the sharing groups that the member has joined in its PIN_FLD_ORDERED_BALGRPS array so that balance impacts are applied to the owning balance group for each sharing group. If the array contains no sharing groups, all balance impacts are applied to the member's balance group.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP uses the element IDs in the PIN_FLD_ORDERED_BALGRPS array to determine the order in which to apply the shared discounts, charges, and products. The array has separate segments for discount sharing groups, charge sharing groups, and product sharing groups. The segment for each one is independent of the segment for the others; BRM does not intermingle the groups. In the array, product sharing groups are always first, followed by discount sharing groups, and then charge sharing groups.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP takes as input an **/ordered_balgrp** object and a string that specifies whether to create, modify, delete, or list the ordered balance group object.

> **Note:**
>
> The **/ordered_balgrp** object must be specified if the action is to modify or delete the object.

If successful, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP returns the POID of the **/ordered_balgrp** object and the POID of the event generated as a result of the creation, modification, or deletion.

If the action is **List**, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP returns all sharing group objects and their ranks in the **/ordered_balgrp** object.

After you create an **/ordered_balgrp** object, you can modify the sequence of the groups in the ordered balance group list. See "Customizing the Order in Which to Apply Sharing Groups".

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP fails in the following cases:

- When the PIN_FLD_ORDERED_BALGROUPS array contains more than one instance of a sharing group.
- When the PIN_FLD_ACTION is **Create**, but an **/ordered_balgrp** object for the account or service already exists.

Creating an Ordered Balance Group

To create an ordered balance group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Creates an **/ordered_balgrp** object for the **/account** or **/service** object specified in the input flist.
2. Adds the sharing group objects specified in the input flist to the **/ordered_balgrp** object.

   This opcode arranges the sharing groups in the order defined by the element IDs in the input flist. The default order is:

   a. Automated product sharing groups
   b. Automated discount offer sharing groups
   c. Discount sharing groups
   d. Charge sharing groups
3. Calls the PCM_OP_SUBSCRIPTION_POL_ORDERED_BALGRP_PRIORITY policy opcode to customize the order. By default, this policy opcode does nothing.
4. Generates an **/event/billing/ordered_balgrp/create** event to record the object creation.

Alternatively, you can use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY to create one or more ordered balance groups. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Adding a Sharing Group to an Ordered Balance Group

To add a sharing group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Adds the sharing group objects specified in the input flist to the **/ordered_balgrp** object.

   This opcode arranges the sharing groups in the order defined by the element IDs in the input flist. The default order is:

   a. Automated product sharing groups
   b. Automated discount offer sharing groups
   c. Discount sharing groups
   d. Charge sharing groups
2. Calls the PCM_OP_SUBSCRIPTION_POL_ORDERED_BALGRP_PRIORITY policy opcode to customize the order. By default, this policy opcode does nothing.

3. Generates an **/event/billing/ordered_balgrp/modify** event to record the object modification.

Alternatively, you can use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY to add a sharing group to one or more ordered balance groups. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Deleting a Sharing Group from an Ordered Balance Group

To delete a sharing group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

> **Caution:**
>
> The element ID for each sharing group in the **/ordered_balgrp** object is unique. If you use an incorrect element ID for the sharing group you want to delete, this opcode deletes the wrong sharing group.

1. Deletes the sharing group objects specified in PIN_FLD_ORDERED_BALGROUPS array from the **/ordered_balgrp** object.

   > **Note:**
   >
   > If the action is **Delete** but the input flist does not contain a PIN_FLD_ORDERED_BALGROUPS array, the **/ordered_balgrp** object is deleted. See "Deleting an Ordered Balance Group".

2. Generates an **/event/billing/ordered_balgrp/modify** event to record the object modification.

Customizing the Order in Which to Apply Sharing Groups

Use the PCM_OP_SUBSCRIPTION_POL_ORDERED_BALGRP_PRIORITY policy opcode to customize the order in which BRM applies a member's sharing groups. By default, the policy opcode does nothing and returns the same output as input.

This policy opcode is called by PCM_OP_SUBSCRIPTION_ORDERED_BALGRP.

Deleting an Ordered Balance Group

To delete an ordered balance group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Deletes the **/ordered_balgrp** object specified in the input flist.
2. Generates an **/event/billing/ordered_balgrp/delete** event to record the object deletion.

Creating and Modifying Multiple Ordered Balance Groups Simultaneously

You can add a sharing group to multiple ordered balance groups simultaneously. To do so, use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY creates one or more ordered balance groups (**/ordered_balgrp** objects) for an account or service.

You can also use this opcode to modify the priority of the sharing groups included in one or more existing ordered balance groups. For discount sharing groups, charge sharing groups, and product sharing groups, the rank controls the order in which the group's balances are impacted by events. For profile sharing groups, the rank is not significant.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY takes as input the POID of the sharing group, an array of one or more members, and a field indicating whether to add the sharing group to the beginning or end of the appropriate segment in the ordered balance group list.

If a service identified in the PIN_FLD_MEMBERS array specifies a type-only service, such as GSM, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY creates or modifies **/ordered_balgrp** objects for all instances for that service type. However, it does not create or modify **/ordered_balgrp** objects for any of the subclass instances. For example, if the input flist specifies the GSM service type, the opcode creates or modifies **/ordered_balgrp** objects for all **/service/telco/gsm** instances. However, it does not create or modify **/ordered_balgrp** objects for any of the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so on.

If successful, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY returns the POID of the sharing group and an array of the events that record the action taken for each ordered balance group in the input flist.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY fails in the following cases:

- When the PIN_FLD_MEMBERS array contains more than one instance of an ordered balance group.
- When the sharing group (PIN_FLD_POID) is already included in some existing **/ordered_balgrp** objects in the PIN_FLD_MEMBERS array.

To create or modify ordered balance groups, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY does the following:

1. Verifies that the input flist contains a PIN_FLD_ORDERED_BALGROUPS array with at least one member.
2. Selects a member in the PIN_FLD_MEMBERS array and determines whether the database already contains an **/ordered_balgrp** object for that member.

   If not, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to create the object. See "Creating an Ordered Balance Group".
3. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to modify the **/ordered_balgrp** object so that it includes the new sharing group.

   See "Adding a Sharing Group to an Ordered Balance Group".
4. Repeats steps 2 through 3 for each remaining **/ordered_balgrp** in the PIN_FLD_MEMBERS array.

Adding and Removing Members Automatically to Sharing Groups

> **Caution:**
>
> Do not call these opcodes directly. Call these opcodes only through the BRM event notification feature. See the following:
>
> - "Configuring Event Notification for Balance Monitoring" in *BRM Managing Customers*.
> - "Configuring Event Notification for Product Sharing Groups" in *BRM Managing Customers*.

BRM uses the following AMS opcodes to manage automated sharing groups:

- PCM_OP_MONITOR_SETUP_MEMBERS to automatically add members to a sharing group when it is first created.

  See "Adding Members to Newly Created Sharing Groups Automatically".
- The following opcodes are used to automatically add or remove members from sharing groups when an account hierarchy or subscription group changes:
  - PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS. See "Adding Members to Hierarchy-Type Balance Monitors".
  - PCM_OP_MONITOR_PROCESS_BILLING_MONITORS. See "Adding Members to Payment Responsibility-Type Balance Monitors".
  - PCM_OP_MONITOR_ACCOUNT_HIERARCHY. See "Adding Members to Hierarchy-Type Sharing Groups Automatically".
  - PCM_OP_MONITOR_BILLING_HIERARCHY. See "Adding Members to Paying Responsibility-Type Sharing Groups Automatically".
  - PCM_OP_MONITOR_SERVICE_HIERARCHY. See "Updating Subscription-Type Monitors Automatically".
  - PCM_OP_MONITOR_HIERARCHY_CLEANUP. See "Removing Members from Balance Monitor Groups".
- PCM_OP_BILL_GROUP_REMOVE_PAYING_MEMBERS to remove all members from an account hierarchy when its parent account is removed. See "Removing Members from Sharing Groups".

Adding Members to Newly Created Sharing Groups Automatically

> **Caution:**
>
> Do not call this opcode directly. This opcode should be called through the event notification feature only.

The PCM_OP_MONITOR_SETUP_MEMBERS opcode automatically adds members to a balance monitor group, product sharing group, or discount sharing group when it is first created. This opcode is triggered by the following events:

- **/event/group/sharing/monitor/create**
- **/event/group/sharing/products/create**
- **/event/group/sharing/discounts/create**

PCM_OP_MONITOR_SETUP_MEMBERS is a wrapper opcode that, according to the group type, calls other standard PCM_OP_MONITOR_* opcodes to:

- Find and add members to the appropriate sharing group (**/group/sharing/monitor**, **/group/sharing/products**, or **/group/sharing/discounts** object).
- Update each member's ordered balance group list (**/ordered_balgrp** object).
- (Balance monitors groups only) Add each member's current balance to the group balance (**/balance_group/monitor** object).

This opcode retrieves the group type and owner, and then calls one of these opcodes:

- **For hierarchy discount sharing group or product sharing group types (H_DSG** or **H_PSG)**: Calls PCM_OP_MONITOR_ACCOUNT_HIERARCHY. See "Adding Members to Hierarchy-Type Sharing Groups Automatically".
- **For hierarchy credit exposure group types (H_CE)**: Calls PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS. See "Adding Members to Hierarchy-Type Balance Monitors".
- **For payment responsibility discount sharing group or product sharing group types (PR_DSG** or **PR_PSG)**: Calls PCM_OP_MONITOR_BILLING_HIERARCHY. See "Adding Members to Paying Responsibility-Type Sharing Groups Automatically".
- **For payment responsibility credit exposure group types (PR_CE)**: Calls PCM_OP_MONITOR_PROCESS_BILLING_MONITORS. See "Adding Members to Payment Responsibility-Type Balance Monitors".
- **For subscription credit exposure group types (SUB_CE)**: Calls PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS.

Adding Members to Hierarchy-Type Balance Monitors

Use the PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS opcode to add members to hierarchy-type balance monitors. This opcode finds and adds the following members to the balance monitor:

- The parent account and its subscriptions
- All *paying* child accounts and their subscriptions
- All *nonpaying* child accounts and their subscriptions

This opcode is called directly by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS performs the following tasks:

1. Finds all services that are associated with the parent account.
2. Finds in the hierarchy all paying and nonpaying child accounts that belong to the parent account.
3. Finds all services that are associated with each child account.

4. Adds the following members to the balance monitor:
   - The parent account and its services.
   - All child accounts and their services.
5. Updates each account's and service's **/ordered_balgrp** object.

Adding Members to Payment Responsibility-Type Balance Monitors

Use the PCM_OP_MONITOR_PROCESS_BILLING_MONITORS opcode to add members to payment responsibility-type balance monitors. This opcode finds and adds the following members to the balance monitor:

- The parent account and its subscriptions
- All *nonpaying* child accounts and their subscriptions

This opcode is called directly by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

PCM_OP_MONITOR_PROCESS_BILLING_MONITORS performs the following tasks:

1. Retrieves the parent's bill unit (**/billinfo**).
2. Searches for all **/billinfo** objects that have PIN_FLD_PARENT_BILLINFO_OBJ as the current bill unit.
3. Searches for the owning account and all services for each bill unit found.
4. Adds the following members to the balance monitor:
   - The parent account and its services
   - All nonpaying child accounts and their services.

   See "Adding Members to Hierarchy-Type Sharing Groups Automatically".
5. Updates each account's and service's **/ordered_balgrp** object.

   See "Adding a Sharing Group to an Ordered Balance Group".

Adding Members to Hierarchy-Type Sharing Groups Automatically

> **Caution:**
>
> Do not call this opcode directly. This opcode should be called through the event notification feature only.

The PCM_OP_MONITOR_ACCOUNT_HIERARCHY opcode finds and adds the following members to hierarchy-type product sharing groups and discount sharing groups:

- The parent account and its subscriptions
- All paying child accounts and their subscriptions
- All nonpaying child accounts and their subscriptions

PCM_OP_MONITOR_ACCOUNT_HIERARCHY also adds members to hierarchy-type balance monitors when an account hierarchy changes, such as when:

- An account is added to the account hierarchy

- A child account adds a service
- The association between a balance group and a bill unit changes
- A line is transferred to an account member

This opcode is triggered by the following events:

- **/event/group/member**
- **/event/notification/service/pre_purchase**
- **/event/audit/subscription/transfer**

PCM_OP_MONITOR_ACCOUNT_HIERARCHY finds all parent accounts at a higher level in the hierarchy than the specified account or service. For each parent account, the opcode performs the following tasks:

1. Finds all sharing groups associated with the parent account.
2. Adds the account or service to the sharing group (**/group/sharing/monitor**, **/group/sharing/products**, or **/group/sharing/discounts** objects).
3. Adds the sharing group to the account's or service's ordered balance group list (**/ordered_balgrp** object).
4. (For Balance Monitor Groups Only) Adds the account or service balance to the monitored balance (**/balance_group/monitor** objects).

Adding Members to Paying Responsibility-Type Sharing Groups Automatically

> **Caution:**
>
> Do not call this opcode directly. This opcode should be called through the event notification feature only.

The PCM_OP_MONITOR_BILLING_HIERARCHY opcode adds members to payment responsibility-type sharing groups, including payment responsible product sharing groups (PR_PSG), payment responsible discount sharing groups (PR_DSG), and payment responsible real-time credit enforcement groups (PR_RTCE).

The opcode adds members to sharing groups when changes occur in an account hierarchy, such as when:

- A nonpaying child account is added to an account hierarchy
- A nonpaying child account adds a service
- A child account changes from a paying to a nonpaying payment type
- The association between a balance group and a bill unit changes
- A line is transferred to a nonpaying child account
- The credit enforcement changes

This opcode is triggered by the following events:

- **/event/notification/service/pre_purchase**
- **/event/customer/billinfo/modify**

- **/event/notification/bal_grp/modify**

PCM_OP_MONITOR_BILLING_HIERARCHY finds all parent accounts at a higher level in the hierarchy than the specified account or service. For each parent account, the opcode performs the following tasks:

1. Finds all payment responsibility-type sharing groups associated with the parent account.
2. Adds the account or service to the group (**/group/sharing/products**, **/group/sharing/monitor**, or **/group/sharing/discounts** object).
3. Adds the sharing group to the account's or service's ordered balance group list (**/ordered_balgrp** object).
4. (Balance Monitor Groups Only) Adds the account or service balance to the balance monitor (**/balance_group/monitor** object).

Updating Subscription-Type Monitors Automatically

> **Note:**
>
> Do not call this opcode directly. This opcode should be called through the event notification feature only.

Use the PCM_OP_MONITOR_SERVICE_HIERARCHY opcode to add members to subscription-type monitors when a subscription group changes. For example, when you add a member service to a subscription, this opcode adds the service to any associated balance monitors.

This opcode is triggered by the **/event/notification/service/pre_purchase** notification event.

PCM_OP_MONITOR_SERVICE_HIERARCHY finds all parent subscription services at a higher level in the group than the specified service. For each parent subscription service, the opcode performs the following tasks:

1. Finds all subscription-type monitors associated with the parent subscription service.
2. Adds the service to the subscription-type monitor groups (**/group/sharing/monitor** objects).
3. Determines whether the added service is a subscription service:
   - If it is, proceeds to the next step.
   - If it is not, adds the monitor groups to the service's **/ordered_balgrp** object.
4. Adds the service balance to the balance monitors (**/balance_group/monitor** objects).

Removing Members from Balance Monitor Groups

> **Note:**
>
> Do not call this opcode directly. This opcode should be called through the event notification feature only.

Use the PCM_OP_MONITOR_HIERARCHY_CLEANUP opcode to remove members from balance monitor groups when an account hierarchy changes. For example, this opcode is called when any of the following actions affect an account hierarchy:

- An account is removed from the hierarchy
- A child account changes from a *nonpaying* to a *paying* payment type
- The association between a balance group and a bill unit changes
- A line is removed from a child account
- The credit enforcement changes

This opcode supports hierarchy credit exposure monitor groups (H_CE), payment responsible real-time credit enforcement monitor groups (PR_RTCE), subscription credit exposure monitor groups (SUB_CE), and payment responsibility credit exposure monitor groups (PR_CE).

This opcode is triggered by the following events:

- **/event/customer/billinfo/modify**
- **/event/group/member**
- **/event/notification/bal_grp/modify**

Removing Members from Sharing Groups

Use the PCM_OP_BILL_GROUP_REMOVE_PAYING_MEMBERS opcode to remove all members from a sharing group when its owner is removed.

This opcode supports only hierarchy product sharing groups (H_PSG) and hierarchy discount sharing groups (H_DSG).

PCM_OP_BILL_GROUP_REMOVE_PAYING_MEMBERS takes as input the **/account** POID of the group's owner and then does the following:

1. Reads the **/group/sharing/products** or **/group/sharing/discounts** object to verify that its PIN_FLD_TYPE_STR field is set to **H_PSG** or **H_DSG**. If the field has another value or is missing, the opcode exits.
2. For each member in the group, calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the sharing group**(/group/sharing/products** or **/group/sharing/discounts** object) from the member's ordered balance group list (**/ordered_balgrp** object).
3. Deletes the sharing group (**/group/sharing/products** or **/group/sharing/discounts** object).
4. Generates an **/event/group/sharing/products/delete** or **/event/group/sharing/discounts/delete** event to record the deletion.

Displaying Balance Monitor Information in Client Applications

You can retrieve and display the following information in your client application:

- The balances for a monitor group
- A list of monitor groups owned by an account or service

Retrieving the Balances for a Monitor Group

To retrieve the total rolled-up balance for a monitor group, call the PCM_OP_BAL_GET_MONITOR_BAL opcode. The opcode retrieves either the balance monitor's current balance or the balance total for a specified time period.

The opcode returns one of the following, depending on whether you pass the PIN_FLD_DATE_BALANCES array in the input flist:

- If you pass the array with a start and end date, the opcode returns the total amount generated by members between those days.

> **Note:**
>
> The end date is exclusive. For example, if you enter January 1 as the start date and January 5 as the end date, the opcode sums all balance impacts generated on January 1, 2, 3, and 4.

- If you do not pass the array, the opcode returns the balance monitor's current rolled-up balance.

This opcode performs the following tasks to retrieve the balances:

1. Retrieves the POID of the **/balance_group/monitor** object and, optionally, the dates for which the balances are requested.
2. Reads the **/balance_group/monitor** object and retrieves the balance information.
3. When the input flist contains dates, retrieves the correct bucket for the specified dates from the sub-balances.
4. Returns the balance for the specified monitor group.

Retrieving the Balance Monitors Owned by an Account or Service

To retrieve a list of balance monitors owned by an account or service, call the PCM_OP_BAL_GET_ACCT_MONITORS opcode.

This opcode takes as input the POID of the **/account** or **/service** object and performs the following tasks:

1. Searches for all the **/group/sharing/monitor** objects associated with the account or service specified in the input flist.
2. For each **/group/sharing/monitor** object associated with the account or service, retrieves the **/balance_group/monitor** objects.
3. Returns a list of monitor groups owned by the specified account or service.

Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications

When you run the **pin_monitor_balance** utility to update monitored balances with the current monitor impacts, it calls the PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode.

PCM_OP_BAL_APPLY_MONITOR_IMPACTS performs the following actions:

1. Updates the specified **/balance_group/monitor** object.
2. Compares the monitored balance in **/balance_group/monitor** with the credit limit and threshold values stored in the **/config/credit_profile** object.
3. If a credit limit or threshold has been breached, generates one of these notification events:
   - **/event/notification/threshold** when the balance crosses above the credit threshold
   - **/event/notification/threshold_below** when the balance falls below the credit threshold

> **Note:**
>
> The opcode generates only one event if multiple thresholds are crossed. However, if multiple monitors cross the thresholds because of a single balance impact, the utility generates an event for each monitor.

Table 13-2 lists the fields in the notification events.

Table 13-2 Notification Event Fields for Monitoring

Field Name	Description
PIN_FLD_RESOURCE_ID	The resource ID.
PIN_FLD_AMOUNT	The balance impact for this event.
PIN_FLD_BAL_GRP_OBJ	The POID of the **/balance_group/monitor** object.
PIN_FLD_PERCENT	The percentage amount crossed.
PIN_FLD_SOURCE_OBJ	Source of the breach, for example, the POID of the event that caused the breach.
PIN_FLD_ALERT_TYPE	The alert type: credit limit or threshold percentage.
PIN_FLD_MONITOR_TYPE	Type of monitor: • Paying responsibility credit exposure • Hierarchy credit exposure • Service level • A type that you have defined.
PIN_FLD_REASON	The reason for the breach: • Upward breach • Downward breach • Upward reset • Downward reset • Unknown reason
PIN_FLD_THRESHOLD	Tracks when the balance of a monitor group crosses a 5% boundary. For example, it tracks whether the current balance is 5%, 10%, or 15% of the credit limit.
PIN_FLD_CREDIT_THRESHOLDS	An array of credit thresholds and limits breached by this event.

Example of Credit Threshold Notification Event Generation

Consider a balance monitor for an account or service with:

- A credit floor of $0

- A credit limit of $100
- Thresholds set at 25%, 75%, and 90%
- A current balance of $50

When a new event comes in with a balance of $26, the new balance is set to $76, and a notification event with the following information is generated:

```
PIN_FLD_ALERT_TYPE    Threshold
PIN_FLD_REASON    Upward breach
PIN_FLD_CREDIT_THRESHOLDS    75%
```

Managing Balance Monitors

To manage balance monitors (**/balance_group/monitor** objects), BRM uses the following opcodes:

- PCM_OP_CUST_SET_BAL_GRP

  See "Creating and Updating Balance Monitors".
- PCM_OP_CUST_DELETE_BAL_GRP

  See "Deactivating Balance Monitors".

Creating and Updating Balance Monitors

Use PCM_OP_CUST_SET_BAL_GRP to create or update a balance monitor (**/balance_group/monitor** object). PCM_OP_CUST_SET_BAL_GRP is a wrapper opcode that performs all necessary tasks to set up the balance monitor and create a link to the customer account.

PCM_OP_CUST_SET_BAL_GRP is called by the PCM_OP_CUST_COMMIT_CUSTOMER opcode during the account creation process, and by the PCM_OP_CUST_MODIFY_CUSTOMER opcode during the account modification process.

To create a **/balance_group/monitor** object, call the PCM_OP_CUST_SET_BAL_GRP opcode with the following information in the input flist:

- The owner of the balance monitor:
  - When the owner is an account, the POID of the **/account** object
  - When the owner is a service, the POIDs of the **/account** and **/service** objects
- Monitor name
- Credit limit
- Credit floor
- Credit thresholds
- Whether to ignore the account's credit limit

> **Note:**
>
> You can also pass this information in the PIN_FLD_ACCTINFO array of the input flist to PCM_OP_CUST_COMMIT_CUSTOMER when creating an account. This opcode automatically calls PCM_OP_CUST_SET_BAL_GRP to create a **/balance_group/monitor** object when the relevant information is in the input flist and balance monitoring is enabled.

PCM_OP_CUST_SET_BAL_GRP does the following:

1. Verifies that balance monitoring is enabled and that, in the PIN_FLD_BAL_INFO input flist array, PIN_FLD_POID is set to a type-only POID for **/balance_group/monitor**.
2. Checks the value of the PIN_FLD_TYPE_STR input flist field in the PIN_FLD_BAL_INFO array:
   - If set to **PR_RTCE**, the opcode creates or updates the balance monitor group (**/group/sharing/monitor** object).
   - If set to **PR_CE** or **H_CE**, the opcode publishes the balance monitor impacts to the monitor queue (**/monitor_queue** object).

Deactivating Balance Monitors

You cannot delete **/balance_group/monitor** objects. You can, however, deactivate the monitor group by deleting its associated **/group/sharing/monitor** object. To do so:

1. Run the **pin_monitor_balance** utility to update balances.
2. Delete the associated **/group/sharing/monitor** object by calling the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode.

   This opcode removes the reference to the **/group/sharing/monitor** object from corresponding **/ordered_balgrp** object.

Managing Balance Groups

To manage balance groups, BRM uses the following opcodes:

- PCM_OP_CUST_SET_BAL_GRP

  See "Creating Balance Groups ".
- PCM_OP_CUST_MODIFY_CUSTOMER

  See "Moving a Balance Group from One Bill Unit to Another".
- PCM_OP_CUST_DELETE_BAL_GRP

  See "Deleting a Balance Group".

Creating Balance Groups

You create balance groups when you create product offerings in PDC or Pricing Center. To create a balance group, BRM uses PCM_OP_CUST_SET_BAL_GRP. This is a wrapper opcode that performs all necessary tasks to set up the **/balance_group** object and create a link to the customer account.

PCM_OP_CUST_COMMIT_CUSTOMER calls PCM_OP_CUST_SET_BAL_GRP when creating a customer account.

PCM_OP_CUST_MODIFY_CUSTOMER calls PCM_OP_CUST_SET_BAL_GRP when modifying customer account information.

If the PIN_FLD_SERVICE_OBJ field in the input flist has a NULL value, PCM_OP_CUST_SET_BAL_GRP creates a balance group associated with the account. If the PIN_FLD_SERVICE_OBJ field specifies a service object POID, a balance group is created and associated with the service.

PCM_OP_CUST_SET_BAL_GRP calls the following opcodes:

- PCM_OP_BILL_SET_LIMIT_AND_CR to set the credit limit in the balance group.
- PCM_OP_CUST_CREATE_BAL_GRP to create the **/balance_group** object.
- PCM_OP_CUST_MODIFY_BAL_GRP to modify an existing **/balance_group** object.

  Based on which field PCM_OP_CUST_MODIFY_BAL_GRP is modifying, the following fields are either mandatory or optional:

  – PIN_FLD_NAME

  – PIN_FLD_BILLINFO_OBJ

  While modifying the balance group, PCM_OP_CUST_SET_BAL_GRP checks if there are any pending bill items. If there are pending bill items and if the billing information has changed, PCM_OP_CUST_SET_BAL_GRP changes the billing information with the new billing information.

Customizing the Default Balance Group of a Bill Unit

In general, BRM applies balance impacts of events to the balance group of the service for which the event was generated. However, some events are not generated for any service. For example, bill adjustment and dispute, payment, payment incentive, and payment fee events are generated for a specific bill unit. In such instances, BRM applies the balance impact of the event to the default balance group of the bill unit and assigns the event to an item belonging to the default service of the default balance group.

BRM chooses a random balance group to be the default for new bill units during account creation and account maintenance.

To create a custom algorithm to select the default balance group of a bill unit, use PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC. You can also use this opcode to select the default service for the default balance group.

This opcode is not called by any opcode. This opcode is an empty hook.

BRM selects a default service using the same algorithm for selecting the default balance group. It selects the service object in the default balance group with the earliest create time (PIN_FLD_CREATED_T). If more than one service object is selected, the service object with the smallest Portal Object ID (POID) is selected as the default service of the default balance group.

> **Note:**
>
> By overriding the BRM algorithm, you change the system behavior that could impact other functionality. For example, if you choose to use only the object creation time or the object POID as the selection criterion, multiple balance group objects might qualify. You should have a good understanding of the default implementation before you customize this opcode.

Moving a Balance Group from One Bill Unit to Another

To move a balance group from one bill unit to another in the same account, BRM uses PCM_OP_CUST_MODIFY_CUSTOMER.

Important:

- You cannot move an account default balance group to a different bill unit.
- You should never move a balance group to a bill unit in a different account. To have a different account be responsible for the charges in a balance group, you must create a bill unit hierarchy that includes the appropriate bill units from both accounts. See "Setting Up Sharing Relationships among Bill Units".

You cannot move a balance group to another bill unit if the balance group's bill unit has unallocated payments or adjustments, open refunds, or unresolved disputes. All disputes must be settled, refunds paid, and payments and adjustments allocated before the balance group can be moved.

A bill unit must have at least one balance group. When a bill unit has only one balance group and that balance group is moved to another bill unit, PCM_OP_CUST_MODIFY_CUSTOMER automatically creates a new balance group not associated with any service for the bill unit from which the balance group was moved.

Deleting a Balance Group

To delete a balance group, use PCM_OP_CUST_DELETE_BAL_GRP.

The POID of the object is checked to ensure that the object can be deleted and that the user has permission to delete the object.

Modifying Sub-balances

Sub-balances are modified when balances are granted or consumed. BRM uses the following opcodes to modify sub-balances. Specify the **/balance_group** object that contains the sub-balance to modify on the input flist:

- PCM_OP_BAL_CHANGE_VALIDITY changes a sub-balance validity period.

  For more information, see "Modifying the Sub-balance Validity Period".
- PCM_OP_BILL_DEBIT debits or credits a noncurrency sub-balance.
- PCM_OP_AR_ACCOUNT_ADJUSTMENT debits or credits a currency sub-balance.
- PCM_OP_BILL_TRANSFER_BALANCE transfers a positive balance from one **/balance_group**. The source and destination balance groups can be in different accounts.

- PCM_OP_BILL_SET_LIMIT_AND_CR sets the credit limit for currency and noncurrency sub-balances.

Modifying the Sub-balance Validity Period

Sub-balance validity periods define when a granted balance is available for consumption. PCM_OP_BAL_CHANGE_VALIDITY updates the valid dates for sub-balances in ***/balance_group*** objects.

To change the validity period of balances, specify the following fields in the PIN_FLD_SUB_BALANCES array on the input flist:

- If you are changing the validity period of more than one sub-balance:
  - Specify the balance element ID in PIN_FLD_RESOURCE_ID.
  - Specify the sub-balance array index in PIN_FLD_ELEMENT_ID.
- If you are setting fixed dates:
  - Specify the new start time in PIN_FLD_VALID_FROM.
  - Specify the new end time in PIN_FLD_VALID_TO.

Converting an Account Hierarchy to Wholesale Billing

Use the PCM_OP_CUST_CONVERT_WHOLESALE_HIERARCHY opcode to convert existing individual account hierarchies to use wholesale billing. See "Converting Existing Bill Unit Hierarchies to Wholesale Billing" in *Configuring and Running Billing* for more information about the conversion.

You set the paying parent bill unit by providing the information in the input flist.

The PCM_OP_CUST_CONVERT_WHOLESALE_HIERARCHY opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Checks for the wholesale configuration, either system-wide or through the business profile configuration.
3. Checks the PARENT_FLAG for the flag indicating wholesale conversion.
4. Checks to ensure that the parent account does not have multiple parent bill units.
5. Calls a stored procedure that does the following:
   a. Finds all the billable item types for the subordinate accounts and creates a corresponding item at the parent level if necessary.
   b. For each item type, gets the POID of the parent-level item and update that value in the PIN_FLD_AR_ITEM_OBJ of the subordinate accounts.
6. Calls the opcode PCM_OP_BILL_UPDATE_JOURNAL with a new flag value PIN_BILL_PROCESS_ONLY_CHILD_JOURNALS. This finds the journals for the subordinate accounts for the current bill cycle, rolls them up into the parent account, and deletes the journals from the subordinate accounts.

> **Note:**
>
> The conversion steps will only consider the current cycle. This feature is not supported for items that have already been billed or AR actions that have already been taken.
>
> When sending input into batch accounts receivable operations, all of the events should be either before wholesale conversion or after conversion. Never send wholesale and non-wholesale events in the same batch operation.
>
> When an event-level dispute is completed before the conversion to wholesale billing, and settlement is completed after the conversion to wholesale billing, then the **/tmp_ar_item_to_roll_up** item will not be generated for the settlement. Instead, an item transfer will be done to transfer the settlement amount to the corresponding billable item.

Rolling Up Non-Currency Balance from Members to the Owner of a Wholesale Sharing Group

Use PCM_OP_SUBSCRIPTION_SHARING_ROLLUP_BALANCE to roll up cumulative non-currency charges from a member's sub-balance to the sharing group owner's balance. This opcode is called by the **pin_rollup_sharing_balances** utility and the PCM_OP_BILL_MAKE_BILL opcode.

The PCM_OP_SUBSCRIPTION_SHARING_GROUP_ROLLUP does the following:

1. Checks if sharing owner group has type rollup enabled. Value is 1 if enabled, 0 if disabled.
2. Retrieves the summary of balance from the **/tmp_sharing_rollup_imp** object and rolls it up to the sharing group owner's balance.
3. Calls PCM_OP_ACT_USAGE in CALC_ONLY mode to publish the **/event/notification/biling/updaterollupbalances** notification event.

14

Installment Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) installment opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Creating Installment Schedule Specifications
- Validating Installment Schedule Specifications
- Preparing Installment Schedule Specifications
- Modifying Installment Schedule Specifications
- Retrieving Installment Schedule Specifications
- Getting Installment Schedule Specifications
- Creating Installments
- Validating Installments
- Preparing Installments
- Applying Installment Charge
- Updating Installment Status
- Retrieving Installments
- Updating Installments
- Canceling Installments
- Installment Proposal

Opcodes Described in This Chapter

Table 14-1 describes the installment management opcodes.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 14-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_INSTALLMENT_APPLY_CHARGE	Applying Installment Charge

Table 14-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_INSTALLMENT_CANCEL_INSTALLMENT	Canceling Installments Validating Installments Customizing How to Bill Events That Occur between Billing Cycles Customizing Credit Limit and Consumption Rules
PCM_OP_INSTALLMENT_CREATE_INSTALLMENT	Creating Installments Validating Installments Installment Proposal Creating Installment Schedule Specifications Validating Installment Schedule Specifications Customizing How to Bill Events That Occur between Billing Cycles Running Bill Now Transferring Balances between Items
PCM_OP_INSTALLMENT_CREATE_SCHEDULE_SPEC	Creating Installment Schedule Specifications Validating Installment Schedule Specifications Preparing Installment Schedule Specifications
PCM_OP_INSTALLMENT_GET_INSTALLMENTS	Retrieving Installments Finding a Bill
PCM_OP_INSTALLMENT_GET_SCHEDULE_SPEC	Retrieving Installment Schedule Specifications Getting Installment Schedule Specifications Validating Installment Schedule Specifications Preparing Installment Schedule Specifications
PCM_OP_INSTALLMENT_MODIFY_SCHEDULE_SPEC	Modifying Installment Schedule Specifications Validating Installment Schedule Specifications Preparing Installment Schedule Specifications Creating Installment Schedule Specifications
PCM_OP_INSTALLMENT_POL_GET_SCHEDULE_SPEC	Getting Installment Schedule Specifications Retrieving Installment Schedule Specifications Creating Installment Schedule Specifications
PCM_OP_INSTALLMENT_POL_PREP_INSTALLMENT	Preparing Installments
PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC	Preparing Installment Schedule Specifications Validating Installment Schedule Specifications
PCM_OP_INSTALLMENT_POL_PROPOSAL	Installment Proposal
PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT	Validating Installments Creating Installments Validating Installment Schedule Specifications
PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC	Validating Installment Schedule Specifications Preparing Installment Schedule Specifications
PCM_OP_INSTALLMENT_PROPOSAL	Installment Proposal

Table 14-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT_STATUS	Updating Installment Status Customizing How to Bill Events That Occur between Billing Cycles Customizing Credit Limit and Consumption Rules
PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT	Updating Installment Status Validating Installments Preparing Installment Schedule Specifications

Creating Installment Schedule Specifications

Use PCM_OP_INSTALLMENT_CREATE_SCHEDULE_SPEC to create an installment schedule specification, which defines the terms for an installment plan your company offers your customers.

You set the installment schedule specification name, start date, end date, minimum installment amount, the maximum number of unequal installments, unequal installment frequency, installment type, and eligibility criteria by passing the information in the input flist.

You specify the eligibility criteria by including the following in the PIN_FLD_CRITERIA input flist array:

- PIN_FLD_CLASS_NAME: The class name, such as **/account** or **/service**
- PIN_FLD_FIELD_NAME: The field's name, such as PIN_FLD_POID or PIN_FLD_FLAGS
- PIN_FLD_OPERATOR: The operator type: equal to (**0**), less than (**1**), greater than (**2**), less than or equal to (**3**), greater than or equal to (**4**), not equal to (**5**), in (**6**), like (**7**), null (**8**), or not null (**9**).
- PIN_FLD_TYPE: The field's data type: integer (**1**), enum (**3**), string (**5**), POID (**7**), timestamp (**8**), or decimal (**14**)
- PIN_FLD_VALUE: The field's value, such as PIN_FLDT_POID or 2

For example, to set the eligibility criteria to accounts that own services like **/service/telco/gsm/telephony** and **/service/telco/gsm/sms**, you would use the following:

```
0 PIN_FLD_CRITERIA          ARRAY [0]
1     PIN_FLD_CLASS_NAME      STR [0] "/service"
1     PIN_FLD_FIELD_NAME      STR [0] "PIN_FLD_POID"
1     PIN_FLD_OPERATOR       ENUM [0] 7    # "PIN_INSTL_OPERATOR_LIKE"
1     PIN_FLD_TYPE           ENUM [0] 7    # PIN_FLDT_POID
1     PIN_FLD_VALUE           STR [0] "/service/telco/gsm"
```

PCM_OP_INSTALLMENT_CREATE_SCHEDULE_SPEC does the following:

- Validates the mandatory parameters in the input.
- Calls PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC to validate the input parameters. See "Validating Installment Schedule Specifications".
- Checks the results of PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC and sends appropriate error messages.

- Calls and gets the output flist from PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC, which it passes as the input flist for any additional functions. See "Preparing Installment Schedule Specifications".

If the object is successfully created, the PCM_OP_INSTALLMENT_CREATE_SCHEDULE_SPEC output flist contains the POID of the **/config/installment/schedule_spec** object.

Validating Installment Schedule Specifications

PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC validates the information in the installment schedule specification (**/config/installment/schedule_spec** object).

PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC performs these validation checks:

- Validates that the installment schedule specification contains the mandatory input parameters.
- If PIN_FLD_TYPE is percentage (**2**), checks that PIN_FLD_MINIMUM is between 0 and 100.
- If PIN_FLD_TYPE is absolute (**1**), checks that PIN_FLD_MINIMUM is 0 or greater.
- Checks that the PIN_FLD_MAX_UNEQUAL_INTERVALS and PIN_FLD_MAX_UNEQUAL_INSTALLMENTS fields are 0 or greater.
- Checks that all elements in the PIN_FLD_CRITERIA array are in the required format.
- Checks that the PIN_FLD_CLASS_NAME field is **/account**, **/service**, or **/profile**.
- If the error buffer is set, clears the error and sets the PIN_FLD_RESULT field to **FAIL**. If the buffer is not set, sets the PIN_FLD_RESULT field to **PASS**.

Preparing Installment Schedule Specifications

Use PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC to prepare installment schedule specification creation fields.

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

Modifying Installment Schedule Specifications

Use PCM_OP_INSTALLMENT_MODIFY_SCHEDULE_SPEC to modify an existing installment schedule specification (**/config/installment/schedule_spec** object).

You can modify the installment schedule specification name, start date, end date, minimum installment amount, the maximum number of unequal installments, unequal installment frequency, installment type, and the eligibility criteria.

PCM_OP_INSTALLMENT_MODIFY_SCHEDULE_SPEC does the following:

- Validates the mandatory input parameters using PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC. See "Validating Installment Schedule Specifications".
- Calls PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC to prepare the input flist and include any customizations. See "Preparing Installment Schedule Specifications".
- Updates the values in the **/config/installment/schedule_spec** object.

Retrieving Installment Schedule Specifications

PCM_OP_INSTALLMENT_GET_SCHEDULE_SPEC retrieves information about the installment schedule specification.

PCM_OP_INSTALLMENT_GET_SCHEDULE_SPEC does the following:

- Validates the mandatory input parameters.
- Gets the **/config/installment/schedule_spec** object from the database.
- Gets the information while creating the installment schedule specifications and the installments.
- During the installment schedule specification creation, gets the status of the installment schedule such as active, inactive, or draft.
- During installment creation, gets only the specifications that match the criteria for the specified account POID.

Getting Installment Schedule Specifications

PCM_OP_INSTALLMENT_POL_GET_SCHEDULE_SPEC is a policy opcode to allow customizations for getting the installment schedule specifications.

Creating Installments

Use PCM_OP_INSTALLMENT_CREATE_INSTALLMENT to create installments for a specific customer.

This opcode retrieves the charge object for the customer and divides the total due amount into the specified number of installments.

Validating Installments

Use PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT to validate the input.

PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT does the following:

- Validates the installment amount. If the amount or percentage is greater than the minimum installment amount or percentage that is defined in the **/config/installment/schedule_spec** object, sends an error.
- Checks for unequal installments in the input PIN_FLD_INSTALLMENTS array and validates them with the value in the INSTALLMENT_SPEC object.
- Validates the sum of all the installments amount with the total installment amount and if not equal, sends an error.
- Checks if the Flags indicate 0 (for the regular bill) or 1 (for the bill now).
- While creating the installments, it checks if the PIN_FLD_STATUS is in the **open** state for the overall installments and individual installments.
- Ensures that the installments are created only for bills or items.
- Validates if the PIN_FLD_START_T is less than PIN_FLD_END_T. If it is, sets the PIN_FLD_RESULT as Pass in the output flist. If not, sets the PIN_FLD_RESULT as Fail in the output flist.

- Validates if the PIN_FLD_START_T is not less than PIN_VIRTUAL_TIME. If it is not, sets the PIN_FLD_RESULT as Pass in the output flist. If it is, sets the PIN_FLD_RESULT as Fail in the output flist.

Preparing Installments

Use PCM_OP_INSTALLMENT_POL_PREP_INSTALLMENT to prepare installment fields.

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

Applying Installment Charge

PCM_OP_INSTALLMENT_APPLY_CHARGE applies the installment charge or amount for the customer on the effective date.

The charge impact of individual installments gets applied to the customer using the Multithreaded Application (MTA) named **pin_installments** after the charge is converted into installments. This MTA gets the installments that must be charged and are due.

PCM_OP_INSTALLMENT_APPLY_CHARGE creates **/event/billing/installment/debit** event and **/item/installment/debit** object, for the installment amount charge applied.

Updating Installment Status

Once an installment is paid, the payment gets allocated, the **/item/installment/debit** object and the **/event/billing/item/transfer** are updated, and PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT_STATUS is called.

PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT_STATUS changes the installment status from **Charged** to **Paid**. Once the installment status is changed to **Paid**, the status of the installment arrangement will be changed to **Finished**.

PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT_STATUS is also called by the PIN_INSTALLMENT_STATUS_CHANGE MTA to change the installment status to Broken if the installment is not paid on the due date.

Retrieving Installments

PCM_OP_INSTALLMENT_GET_INSTALLMENTS retrieves information about all the installments associated with an account and service. If service is not mentioned, retrieves all the installments associated with the account.

PCM_OP_INSTALLMENT_GET_INSTALLMENTS does the following:

- Validates the mandatory parameters in the input flist.
- Gets the **/installment** objects for the account POID.
- Includes PIN_FLD_BILLINFO_OBJ as search criteria, in the search template, if PIN_FLD_BILLINFO_OBJ is present in the input flist.
- Searches for installments in the database based on the input parameters and returns all the installments details.

Updating Installments

PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT updates OPEN installments.

Note: Only open installments can be updated.

If the amount in the installments array is not the same as the sum of the remaining open amount, the opcode returns an error message.

For example, an account has a $300 X 6 months installments hence $1800 amount must be paid. The installment has been paid for two months that is $600 is paid and the remaining amount to be paid is $1200.

The update installment request must not have a different amount in the installments array and the remaining open amount. Both the amount, as in this example, must be $1200. If the amount is different, it sends an error.

PCM_OP_INSTALLMENT_UPDATE_INSTALLMENT does the following:

- Validates the mandatory input parameters in the input flist.
- Validates if the due amount meets the minimum installment due amount as per the installment schedule specification.
- Gets the installment total amount due, adds up the remaining installments, and validates them to ensure the total due amount is equal to the sum of the installment amount.
- Validates the number of remaining installments, the term, and installment elements.
- Calls and gets results from PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT. Sends error message in case of any error.
- Reads the **/installment** object using PCM_OP_READ_OBJ opcode.
- Calls PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC.
- Updates the **/installment** object with the input parameter values.

Canceling Installments

Use PCM_OP_INSTALLMENT_CANCEL_INSTALLMENT to sum any remaining amounts, charge the customer, and cancel the remaining installments.

PCM_OP_INSTALLMENT_CANCEL_INSTALLMENT does the following:

- Validates the mandatory input parameters in the input flist.
- Calls PCM_OP_BILL_SET_LIMIT_AND_CR to update the credit limit to increase the open installments amount and the same will be reverted after creating a one-off charge.
- Gets the balance group for the given installment POID from the **/installment** object.
- Calls PCM_OP_ACT_USAGE to create a one-off charge with the total installment due, for the installment charge with the **/event/billing/installment/debit** event and associates it with the installment charge item.
- Calls PCM_OP_ACT_USAGE to create notification event for the canceled installment, with the **/event/installment/cancel** event.
- Updates the **/installment** object with the input parameter values. Updates the status in the **/installment** object and the status in the **installments_details** array to **canceled**.

Installment Proposal

PCM_OP_INSTALLMENT_PROPOSAL is called by the API layer to prepare the installment proposal.

The opcode refers to the input parameters such as installment amount, installment term (the number of months), and start date and divides the installment amount into equal installments. If the installment amount cannot be divided into equal installments, the opcode rounds it off to the nearest amount and adjusts any remaining amount to the last installment.

PCM_OP_INSTALLMENT_PROPOSAL returns the proposed installment calculated details along with the installment amount and effective dates.

Use the PCM_OP_INSTALLMENT_POL_PROPOSAL policy opcode to modify and customize the return flist of PCM_OP_INSTALLMENT_PROPOSAL.

15

Invoice Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) invoice opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Setting the Default Payment Due Date
- Making Invoices
- Displaying Invoices
- Defining the Invoice Type
- How Invoices Are Formatted
- Including Custom Data in Invoices
- Including the Time Zone in Invoices
- Adding Soft Descriptors to Invoices
- Specifying Event Fields to Cache for Invoicing
- Decoding Cached Event Data for Invoicing
- How Invoices Are Generated
- Customizing Invoice Search Operations
- Generating Trial Invoices

Opcodes Described in This Chapter

Table 15-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 15-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_SPEC_EVENT_CACHE	Including Custom Data in Invoices Specifying Event Fields to Cache for Invoicing
PCM_OP_AR_GET_ALLOCATED_AR_ITEMS	How Invoices Are Generated

Table 15-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_MAKE_BILL	How Trial Invoicing Works for Hierarchical Bill Units in Oracle Analytics Publisher
PCM_OP_BILL_MAKE_TRIAL_BILL	Generating Trial Invoices
PCM_OP_CUST_COMMIT_CUSTOMER	Defining the Invoice Type
PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE	About Associating Bill Units with an Oracle Analytics Publisher Invoice and Report About the /associated_bus_profile Object
PCM_OP_CUST_CREATE_BILLINFO	About Associating Bill Units with an Oracle Analytics Publisher Invoice and Report About the /associated_bus_profile Object
PCM_OP_CUST_POL_PREP_PAYINFO	Setting the Default Payment Due Date
PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE	About Associating Bill Units with an Oracle Analytics Publisher Invoice and Report
PCM_OP_CUST_SET_PAYINFO	Defining the Invoice Type
PCM_OP_INV_DECODE_INVOICE_DATA	Decoding Cached Event Data for Invoicing Creating Custom Search Templates
PCM_OP_INV_FORMAT_INVOICE	How Invoices Are Formatted Customizing the Invoice Format by Using an XSL Style Sheet
PCM_OP_INV_MAKE_INVOICE	Making Invoices Defining the Invoice Type How Invoices Are Formatted
PCM_OP_INV_MAKE_INVOICE	How Invoices Are Generated Customizing Invoice Search Operations Creating Custom Search Templates Configuring Error Checking for Customized Invoicing Generating Trial Invoices
PCM_OP_INV_POL_FORMAT_INVOICE	Customizing the Format for Printed Invoices
PCM_OP_INV_POL_FORMAT_INVOICE	Including Custom Data in Invoices
PCM_OP_INV_POL_FORMAT_INVOICE_DOC1	Customizing the Format for DOC1 Invoices
PCM_OP_INV_POL_FORMAT_INVOICE_HTML	Customizing the Format for HTML Invoices Customizing the Layout of Bill Items in Invoices Customizing the Format for Printed Invoices Including the Time Zone in Invoices
PCM_OP_INV_POL_FORMAT_INVOICE_XML	Customizing the Format for XML Invoices
PCM_OP_INV_POL_FORMAT_INVOICE_XSLT	Customizing the Invoice Format by Using an XSL Style Sheet How Invoices Are Formatted Displaying an Invoice On Demand

Table 15-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_INV_POL_FORMAT_VIEW_INVOICE	Displaying Invoices How Invoices Are Formatted Customizing the Format for HTML Invoices Customizing the Format for XML Invoices Displaying an Invoice On Demand
PCM_OP_INV_POL_POST_MAKE_INVOICE	How Invoices Are Generated Customizing Invoice Search Operations Example: Generating Invoices Based on Event Types Configuring Error Checking for Customized Invoicing
PCM_OP_INV_POL_PREP_INVOICE	How Invoices Are Formatted Customizing the Layout of Bill Items in Invoices Customizing the Format for Printed Invoices
PCM_OP_INV_POL_PREP_INVOICE	Including Custom Data in Invoices
PCM_OP_INV_POL_PREP_INVOICE	Changing the Trial Billing Watermark
PCM_OP_INV_POL_PREP_INVOICE	How Trial Invoicing Works for Hierarchical Bill Units in Oracle Analytics Publisher
PCM_OP_INV_POL_SELECT	Decoding Cached Event Data for Invoicing How Invoices Are Generated Customizing Invoice Search Operations Creating Custom Search Templates Example: Generating Invoices Based on Event Types
PCM_OP_INV_VIEW_INVOICE	Displaying Invoices Displaying an Invoice On Demand
PCM_OP_PYMT_POL_PRE_COLLECT	Adding Soft Descriptors to Invoices

Setting the Default Payment Due Date

The default payment due date for invoice payments is 30 days from the billing date. To change the number of days, customize the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode.

Making Invoices

PCM_OP_INV_MAKE_INVOICE creates an invoice for a specified (regular or corrective) bill object.

For regular invoices, PCM_OP_INV_MAKE_INVOICE does the following:

- Uses the PIN_FLD_INV_DETAIL_FLAG value in the **/payinfo** object to determine whether to generate a detailed invoice or a summary invoice
- Uses the invoicing threshold parameters in the **/config/business_params** object to determine whether the invoices of nonpaying bills in a bill unit (**/billinfo** object) hierarchy should be generated separately or consolidated into the invoice of the paying parent bill

For proforma invoices, PCM_OP_INV_MAKE_INVOICE verifies the bill is in the **InProgress** state and uses the PIN_FLD_MESSAGE value in the **/invoice** object to set the text of the

watermark for the invoice. PCM_OP_INV_MAKE_INVOICE is the initial opcode that gets called to create an invoice for a designated bill object.

PCM_OP_INV_MAKE_INVOICE identifies corrective bills by the entries PIN_OBJ_NAME_CORRECTIVE_BILL, PIN_OBJ_NAME_CORRECTIVE_BILL_NOW, and PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND in the PIN_FLD_NAME field in the bill object.

For corrective invoices, the default value for the type of corrective invoice (whether it is a **Replacement Invoice** or an **Invoice Correction Letter** and whether that document is a summary or in detail) is obtained from the PIN_FLD_INV_TYPE field from the **/event/billing/corrective_bill** object. For invoices in a bill unit hierarchy, the default setting is determined by the corresponding value in the **/event/billing/corrective_bill** object in the parent invoice.

If PIN_FLD_INV_FLAGS is present in the input flist when PCM_OP_INV_MAKE_INVOICE is called, the opcode discards the default value and uses the PIN_FLD_INV_FLAGS value from the input flist with the entries in the **pin_inv.h** file to arrive at the required type of corrective invoice.

The opcode retrieves the correction reason and previous bill number from the **/event/billing/corrective_bill** object and inserts them in the corrective invoice. It calls the **fm_inv_get_previous_bill** to retrieve the contents of the latest **history_bills** object.

Typically, you generate invoices automatically as part of running the **pin_bill_day** billing script. This script runs several utilities, including **pin_inv_accts**, the invoicing utility.

pin_inv_accts runs twice. In the first run, it performs the following tasks to handle bill unit hierarchies:

1. Searches for all bills that have reached the end of their billing cycle and for which invoices have not yet been generated.
2. Checks the nonpaying threshold values in the **/config/business_params** object to determine the maximum number of nonpaying child bill units that are allowed in the bill unit hierarchy.
3. For each **/bill** object retrieved, checks the PIN_FLD_AR_HIERARCHY_SIZE value to determine if it exceeds the threshold value.
4. If the threshold is exceeded, uses multiple threads to retrieve the nonpaying bill units and to generate an invoice for each one.

> **Note:**
>
> If bill suppression is enabled on an account with a paying parent bill unit and the nonpaying child bill unit threshold is exceeded, invoicing fails. This occurs because nonpaying bill units, which use the bill number of their paying parent bill unit's parent account, are generated even when billing for the parent account is suppressed. In such cases, they do not contain bill numbers, which invoices require. To exclude invoicing for these bills, run **pin_inv_accts** with the **-skip_blank_billnos** parameter. (Nonpaying bill units contain bill numbers only when their paying parent bill unit's parent accounts are billed.)

If no errors occur, **pin_inv_accts** runs a second time to generate invoices for parent accounts in hierarchies and for nonhierarchical accounts:

5. Searches for all bills that have reached the end of their billing cycle and for which invoices have not yet been generated.

6. For each **/bill** object retrieved, generates an invoice.

Displaying Invoices

To display invoices, use PCM_OP_INV_VIEW_INVOICE.

This opcode uses the POID of the **/bill** object or **/invoice** object to locate and retrieve a specific invoice. Specify the output format of the invoice as a mime type in the PIN_FLD_TYPE_STR field in the input flist. If you provide the bill number in PIN_FLD_BILL_NO, the opcode searches the bill database and, if necessary, the **history_bills** objects.

The PIN_FLD_FLAGS value in the output flist determines the type of invoice to view, for example, a summary or detailed invoice for a nonhierarchical bill unit, and the PIN_FLD_INV_SIZE value in the output flist specifies the size of the invoice returned.

Specify the output format of the invoice as a mime type in the PIN_FLD_TYPE_STR field on the input flist.

PCM_OP_INV_VIEW_INVOICE performs the following tasks:

- Uses the PIN_FLD_THRESHOLD_UPPER value in the input flist to determine the maximum size (KB) of the invoice to be viewed. If the invoice is larger than the size specified, an error message and the invoice size, rather than the invoice itself, is displayed.
- Checks the following fields in the **/invoice** object:
  - PIN_FLD_INV_SIZE to determine the size of the invoice.

    If the invoice size is greater than the invoicing threshold value, PCM_OP_INV_VIEW_INVOICE does not try to format the invoice.

    If the invoice size is less than the invoicing threshold, PCM_OP_INV_VIEW_INVOICE continues processing.

    To set the maximum size in kilobytes of invoices that can be sent by email, use the **inv_send_size** parameter in the **pin_inv_send** utility. This entry is used by PCM_OP_INV_VIEW_INVOICE to restrict sending large invoices to the Email Data Manager (**dm_email**).
  - PIN_FLD_INV_FLAGS to determine the type of invoice to view.
  - PIN_FLD_TYPE_STR to determine whether the format specified is stored in the object. Invoices can be stored in the database in pin_flist, XML, HTML, or DOC1 format.

    If the specified format is stored, the invoice is retrieved and returned in the format specified. If the specified format is not stored, PCM_OP_INV_POL_FORMAT_VIEW_INVOICE is called to attempt to format the invoice. Invoices can use HTML, XML, or DOC1 format. To apply an XSL style sheet to the invoice, specify the XSL mime type in the PIN_FLD_TYPE_STR field.

If successful, PCM_OP_INV_VIEW_INVOICE returns a buffer containing the formatted invoice. The PIN_FLD_RESULT value in the output flist is **1**.

If not successful due to system errors, no invoice is returned. The PIN_FLD_RESULT value is **0**.

If not successful because the invoice exceeded the threshold size, the PIN_FLD_RESULT value is **2**.

Defining the Invoice Type

The PIN_FLD_INV_TYPE field in the **/payinfo** object defines which type of invoice to generate for a given bill unit. Table 15-2 lists the possible values for PIN_FLD_INV_TYPE and the corresponding type of invoice that BRM generates.

Table 15-2 PIN_FLD_INV_TYPE Values and Invoice Generated by BRM

Value	Binary Notation	Regular Invoice (if) Generated	Corrective Invoice (if) Generated
0	0000 0000	Detail	Detail Replacement Invoice
1	0000 0001	Summary	Detail Replacement Invoice
2	0000 0010	Detail	Summary Replacement Invoice
3	0000 0011	Summary	Summary Replacement Invoice
4	0000 0100	Detail	Detail Invoice Correction Letter
5	0000 0101	Summary	Detail Invoice Correction Letter
6	0000 0110	Detail	Summary Invoice Correction Letter
7	0000 0111	Summary	Summary Invoice Correction Letter

Note:

For regular invoices, the **-detail** and **-summary** parameters of the **pin_inv_accts** utility override the PIN_FLD_INV_TYPE field values in the **/payinfo** object.

For corrective invoices, the **-corr_type**, **-detail**, and **-summary** of the **pin_inv_accts** utility override the PIN_FLD_INV_TYPE field values in the **/payinfo** object.

To generate summary invoices, change the value of the PIN_FLD_INV_TYPE field in the **/payinfo** object to the required value listed in Table 15-2.

- To set this value when creating a customer account, pass it in the input flist of the PCM_OP_CUST_COMMIT_CUSTOMER opcode.
- To set this value when adding or changing a payment method, pass it in the input flist of the PCM_OP_CUST_SET_PAYINFO opcode.

When this value is set, the PCM_OP_INV_MAKE_INVOICE opcode uses it in conjunction with the position of the bill, in relation to other bills, to set the PIN_FLD_INV_FLAGS value in the **/invoice** object. Possible values are listed in Table 15-3:

Table 15-3 PIN_FLD_INV_FLAGS Values

Type of invoice	PIN_FLD_INV_FLAGS Hex value
Summary invoice for a nonhierarchical account.	0x0006
Detailed invoice for a nonhierarchical account.	0x0005
Summary invoice for a nonpaying bill unit.	0x000A

Table 15-3 (Cont.) PIN_FLD_INV_FLAGS Values

Type of invoice	PIN_FLD_INV_FLAGS Hex value
Detailed invoice for a nonpaying bill unit.	0x0009
Summary invoice for a parent A/R account. (Nonconsolidated: does not include nonpaying bill unit data).	0x0012
Detailed invoice for a parent A/R account. (Nonconsolidated: does not include nonpaying bill unit data).	0x0011
Summary invoice for a parent A/R account. (Consolidated: includes nonpaying bill unit data).	0x0001A
Detailed invoice for a parent A/R account. (Consolidated: includes nonpaying bill unit data).	0x00019

For information on generating invoices, see "Making Invoices".

How Invoices Are Formatted

To format invoices, use PCM_OP_INV_FORMAT_INVOICE. This opcode performs XSL transformation on an invoice.

PCM_OP_INV_FORMAT_INVOICE is called by PCM_OP_INV_POL_FORMAT_INVOICE_XSLT to apply an XSL style sheet to an invoice. It receives as input an XML formatted invoice and an XSL style sheet. It applies the style sheet to the invoice and returns the formatted invoice.

> **Note:**
>
> The XSL transformation is done in the Java Server with the help of XSLT software.

PCM_OP_INV_FORMAT_INVOICE uses the POID of the **/bill** object or **/invoice** object to locate and retrieve a specific invoice. Specify the output format of the invoice as a mime type in the PIN_FLD_TYPE_STR field in the input flist.

PCM_OP_INV_FORMAT_INVOICE performs the following tasks:

- Checks the **/invoice** object to see whether the format specified in the PIN_FLD_TYPE_STR field is stored in the object. Invoices can be stored in the database as pin_flist, XML, HTML, or DOC1 format.
- If the specified format is stored, the invoice is retrieved and returned in the format specified.
- If the specified format is not stored, PCM_OP_INV_POL_FORMAT_VIEW_INVOICE is called to attempt to format the invoice. Invoices can use HTML, XML, or DOC1 format. To apply an XSL style sheet to the invoice, specify the XSL mime type in the PIN_FLD_TYPE_STR field.

If successful, PCM_OP_INV_FORMAT_INVOICE returns a buffer containing the formatted invoice. The PIN_FLD_RESULT field in the output flist is set to **1**.

If unsuccessful, no invoice is returned. The PIN_FLD_RESULT field is set to **0**.

Use PCM_OP_INV_POL_PREP_INVOICE to customize invoice formatting. This opcode is called by PCM_OP_INV_MAKE_INVOICE.

For corrective invoicing, PIN_FLD_PREV_BILLINFO field contains the details of previous bills. Additionally, the PIN_FLD_CORRECTION_INFO array under PIN_FLD_OTHER_ITEMS contains all the special items that correspond to the corrective bill and its associated events.

The contents of value in the PIN_FLD_INV_TYPE field determines the contents of the invoice. For example, the invoice for a Detail Replacement shows all the items and events while a Detail Correction letter displays only the special items and its associated. However, both Replacement and Correction Letter invoices show the previous amount, adjusted amount and new amounts for events and items.

Customizing the Layout of Bill Items in Invoices

You can customize the layout of bill items on an invoice.

To list bill items in a different order, use the PCM_OP_INV_POL_PREP_INVOICE policy opcode.

When corrective invoicing is enabled, PIN_FLD_PREV_BILLINFO contains the details of the previous bill (if any). Additionally, the PIN_FLD_CORRECTION_INFO array under PIN_FLD_OTHER_ITEMS contains all the items specific to the corrective bill and its corresponding events.

To make other changes to the layout:

- If you are using an HTML invoice template, use the PCM_OP_INV_POL_FORMAT_INVOICE_HTML policy opcode.
- If you are using an XSLT style sheet for the invoice template, you can perform this level of customization in the style sheet.

Customizing the Format for Printed Invoices

To format invoices for printing, use PCM_OP_INV_POL_FORMAT_INVOICE.

This opcode is called when invoices are generated to specify if the invoices are to be stored in XML or pin_flist format in the **/invoice** object. The default is XML.

- If the storage format is specified as XML or pin_flist, the return flist contains only the specified format type.
- If you customize PCM_OP_INV_POL_FORMAT_INVOICE to generate other formats, the return flist contains the format type along with a buffer containing the formatted invoices.

Any customization of invoice content done by PCM_OP_INV_POL_PREP_INVOICE is passed to PCM_OP_INV_POL_FORMAT_INVOICE prior to storage.

You can customize PCM_OP_INV_POL_FORMAT_INVOICE to generate other storage formats. For example, to store invoices in HTML format, you can add code to call PCM_OP_INV_POL_FORMAT_INVOICE_HTML and then add the formatted invoice to the buffer in the output flist.

Customizing the Format for HTML Invoices

To format invoices in HTML, use PCM_OP_INV_POL_FORMAT_INVOICE_HTML.

This opcode is called by PCM_OP_INV_POL_FORMAT_INVOICE when the invoice format requested is HTML.

If your system has the invoicing-by-service feature enabled, PCM_OP_INV_POL_FORMAT_INVOICE_HTML will display the invoice items by service instance. The **/config/invoice_events** object belonging to the root account is cached.

Customizing the Format for XML Invoices

To format invoices in XML, use PCM_OP_INV_POL_FORMAT_INVOICE_XML.

This opcode is called by PCM_OP_INV_POL_FORMAT_VIEW_INVOICE when the invoice format requested is XML.

Customizing the Invoice Format by Using an XSL Style Sheet

To format invoices using an XSL style sheet, use PCM_OP_INV_POL_FORMAT_INVOICE_XSLT.

To specify the use of XSL style sheets, set the PIN_FLD_FLAGS field in the **/config/invoice_templates** object to **1**.

This opcode is called by PCM_OP_INV_POL_FORMAT_INVOICE when the **/config/invoice_templates** object specifies an XSL style sheet.

Customizing the Format for DOC1 Invoices

To format invoices for DOC1, use PCM_OP_INV_POL_FORMAT_INVOICE_DOC1.

> **Note:**
>
> For DOC1 format, you must have the DOC1 software.

This opcode also checks the **pin.conf** file to see whether your system has service-centric invoicing turned on. If invoicing by service is enabled, this opcode reorganizes the flist so that it displays the invoice items by service instance.

If your system has the invoicing-by-service feature enabled, PCM_OP_INV_POL_FORMAT_INVOICE_DOC1 displays the invoice items by service instance.

Displaying an Invoice On Demand

To display an invoice on demand, use PCM_OP_INV_POL_FORMAT_VIEW_INVOICE. This opcode generates an invoice in the specified format.

BRM can store the invoice in the database in either pin_flist or XML format. The default is pin_flist. The storage format is specified by PCM_OP_INV_FORMAT_VIEW_INVOICE.

PCM_OP_INV_POL_FORMAT_VIEW_INVOICE is called when PCM_OP_INV_VIEW_INVOICE requests an invoice in a format that is not stored on the **/invoice** object. PCM_OP_INV_POL_FORMAT_VIEW_INVOICE attempts to generate the invoice in the requested format. Invoices may be formatted as HTML or DOC1. An XML format is also available, but it displays as HTML format.

PCM_OP_INV_POL_FORMAT_VIEW_INVOICE calls one of the following policy opcodes, depending on the requested format:

- PCM_OP_INV_POL_FORMAT_INVOICE_HTML
- PCM_OP_INV_POL_FORMAT_INVOICE_DOC1
- PCM_OP_INV_POL_FORMAT_INVOICE_XML
- PCM_OP_INV_POL_FORMAT_INVOICE_XSLT

PCM_OP_INV_POL_FORMAT_VIEW_INVOICE checks the PIN_FLD_FLAG field in the **/config/invoice_templates** object to see whether an XSL style sheet should be applied to the invoice. If the flag is set, PCM_OP_INV_POL_FORMAT_INVOICE_XSLT is called to format the invoice.

PCM_OP_INV_POL_FORMAT_VIEW_INVOICE also checks the **pin.conf** file to see whether your system has service-centric invoicing turned on. If invoicing by service is enabled, PCM_OP_INV_POL_FORMAT_VIEW_INVOICE reorganizes the flist so that it displays the invoice items by service instance.

> **Note:**
>
> Because the flist is reorganized, if you apply an XSL style sheet to the invoice, you may need to change the style sheet to reflect the change in output created by the service instance organization.

Including Custom Data in Invoices

To customize the data displayed in invoices, use the PCM_OP_INV_POL_PREP_INVOICE policy opcode.

This policy opcode searches the database for information about the account and creates an flist of relevant fields to include at invoicing time.

You can customize invoice information by performing additional data searches, or modifying large invoice flists. Customization done by the PCM_OP_INV_POL_PREP_INVOICE policy opcode is passed to the PCM_OP_INV_POL_FORMAT_INVOICE policy opcode, where the storage format is specified.

You can customize the events listed on an invoice as follows:

- **Include noncurrency events.** By default, BRM invoices list all events with a currency balance impact greater than zero. Use the PCM_OP_INV_POL_PREP_INVOICE policy opcode to include noncurrency events. For example, you can list hours of usage if you track hours as part of a promotion.
- **List fewer or more events.** Use the PCM_OP_INV_POL_PREP_INVOICE policy opcode.

> **Note:**
>
> Using the PCM_OP_INV_POL_PREP_INVOICE policy opcode to control which event information is displayed on invoices at invoicing time can affect performance if you access information normally not stored in the PIN_FLD_INVOICE_DATA field of the event. You will get better performance if you define the event information to be stored for invoicing when the event is created by using the PCM_OP_ACT_POL_SPEC_EVENT_CACHE policy opcode.

You must modify this policy opcode to enable messaging through the Universal Messaging Service (UMS).

Changing the Trial Billing Watermark

To distinguish the trial invoice from the regular invoice, a **Trial Invoice** watermark is added to all the pages after the first page. You can change the watermark text by editing the PIN_FLD_MESSAGE field's value in the PCM_OP_INV_POL_PREP_INVOICE flist.

Including the Time Zone in Invoices

By default, invoices do not show the customer's time zone. To show the time zone on an invoice, do one of the following:

- If you are using an HTML invoice template, use the PCM_OP_INV_POL_FORMAT_INVOICE_HTML policy opcode.
- If you are using an XSLT invoice template, edit the style sheet to use the PIN_FLD_TIMEZONE_ADJ_END_T and PIN_FLD_RATED_TIMEZONE_ID fields instead of the PIN_FLD_END_T field for the events.

Adding Soft Descriptors to Invoices

Customize PCM_OP_PYMT_POL_PRE_COLLECT to add soft descriptor information to an invoice.

> **Note:**
>
> You can also customize PCM_OP_PYMT_POL_PRE_COLLECT to set a minimum amount to charge.

This example shows how to retrieve the merchant name and a package descriptor. The maximum entry is 22 characters including spaces. If the information is longer than 22 characters, it is truncated on the statement.

```
pin_flist_t*sub_flistp = NULL;
void*vp = NULL;

/*
 * For each element in the PIN_FLD_CHARGES array add the soft
 * descriptors.
 */
```

```
/*
 * Add the merchant info substruct for the soft descriptors.
 */

sub_flistp = PIN_FLIST_SUBSTR_ADD(flistp, PIN_FLD_MERCHANT_INFO, ebufp);

/*
 * Add the merchant "doing business as" to the soft descriptors.
 */

vp = (void *) "Broadband Service";
PIN_FLIST_FLD_SET(sub_flistp, PIN_FLD_MERCHANT, vp, ebufp);

/*
 * Read the charge offer information from the account product array to
 * pick up the /plan object.
 * (Exercise left to reader.)
 */

/*
 * Read package description from the /plan object
 */
vp = (void *) NULL;
s_flistp = PIN_FLIST_CREATE(&ebuf);
vp = PIN_FLIST_FLD_GET(prod_flistp, PIN_FLD_PLAN_OBJ, 0, ebufp);
PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_POID, vp, ebufp);
PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_DESCR, NULL, ebufp);

/*
 * Read plan info.
 */
PCM_OP(ctxp, PCM_OP_READ_FLDS, 0, s_flistp, &r_flistp, ebufp);
vp = PIN_FLIST_FLD_GET(r_flistp, PIN_FLD_DESCR, 0, ebufp);

/*
 * Add the product name to the soft descriptors.
 */

PIN_FLIST_FLD_SET(sub_flistp, PIN_FLD_PROD_NAME, vp, ebufp);

/*
 * Add the merchant customer service phone number to the soft
 * descriptors.
 */

vp = (void *) "800-PORTAL1"
PIN_FLIST_FLD_SET(sub_flistp, PIN_FLD_PHONE, vp, ebufp);
```

Specifying Event Fields to Cache for Invoicing

You can improve performance by limiting the amount of information cached. However, when retrieving information, it is quicker to read from a cached field than from the event table.

The PIN_FLD_INVOICE_DATA field in the ***/event*** object contains a cache of all fields that need to be handled in invoicing. Use PCM_OP_ACT_POL_SPEC_EVENT_CACHE to define which balance impact fields to cache for invoicing.

By default, the opcode caches the following PIN_FLD_BAL_IMPACTS array fields shown in Table 15-4 in the base table field PIN_FLD_INVOICE_DATA.

> **Note:**
>
> The PIN_FLD_INVOICE_DATA field is limited to 4000 bytes. If the event cache size of the PIN_FLD_INVOICE_DATA field is greater than 4000 bytes, it is ignored, and the invoice displays a **0** amount. For Oracle databases, you can increase the size of the **invoice_data** column to work around this limitation.

Table 15-4 Cached PIN_FLD_BAL_IMPACTS Fields

Opcode	Description
PIN_FLD_AMOUNT	The account balance impact. The value can be positive or negative.
PIN_FLD_DISCOUNT	The discount applied to the balance impact.
PIN_FLD_IMPACT_TYPE	Balance impact type - rated by BRM rated-engine (0x1), pre-rated (0x2), taxed (0x4), purchase order (0x8), re-rated(0x20), and reverse_rated(0x40).
PIN_FLD_ITEM_OBJ	Link to the item object affected by this event. Applies only to the balance array element that impacts currency balances. (This may be different from the PIN_FLD_ITEM_OBJ field in the base **/event** object.)
PIN_FLD_QUANTITY	The quantity applied. The number of units actually applied using this pricing.
PIN_FLD_RATE_TAG	Description of the charge pricing used. Same as PIN_FLD_DESCR in the **/rate** object.
PIN_FLD_RESOURCE_ID	Numeric value of the balance element that is impacted.
PIN_FLD_TAX_CODE	Tax code for the rate used. When taxes do not apply, this field is set to **0**.

If you remove these fields from PCM_OP_ACT_POL_SPEC_EVENT_CACHE and leave the event cache turned on, there are no event details in the invoices.

> **Note:**
>
> If you do not use event caching, return a NULL pointer to the caller of PCM_OP_ACT_POL_SPEC_EVENT_CACHE.

You can customize PCM_OP_ACT_POL_SPEC_EVENT_CACHE to cache additional balance impact array fields.

If you disable caching, these fields are read directly from the event table, which slows performance.

> **Note:**
>
> - If you remove the default fields of the PIN_FLD_BAL_IMPACTS array from PCM_OP_ACT_POL_SPEC_EVENT_CACHE and leave the event cache turned on, there are no event details in the invoices.
> - The event cache must not exceed 4000 bytes. If you have a large number of elements in the PIN_FLD_BAL_IMPACTS array, you must disable the **EventCache** business parameter.

To enable or disable caching of the PIN_FLD_BAL_IMPACTS array:

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

   ```
   pin_bus_params -r BusParamsInvoicing bus_params_Invoicing.xml
   ```

   This command creates an XML file named **bus_params_Invoicing.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_Invoicing.xml.out**, set the following entry:

   ```
   <EventCache>value</EventCache>
   ```

   where *value* is one of the following:

   - **enabled**: Specifies to enable event caching.
   - **disabled**: Specifies to disable event caching.

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

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

   ```
   pin_bus_params bus_params_Invoicing.xml
   ```

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

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

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

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Decoding Cached Event Data for Invoicing

PCM_OP_INV_DECODE_INVOICE_DATA parses the event data being retrieved for an invoice. This opcode retrieves the contents of the PIN_FLD_INVOICE_DATA field, parses the data, and returns the decoded data in the output flist.

If you customized PCM_OP_INV_POL_SELECT to search for custom event data, you must call PCM_OP_INV_DECODE_INVOICE_DATA.

The input flist contains the PIN_FLD_INVOICE_DATA field, which is a cached string that needs to be decoded. It is limited to 4000 bytes. If the cache size is greater than 4000 bytes, it is ignored.

The output flist contains the PIN_FLD_EXTENDED_INFO substruct, which contains the fields decoded from the PIN_FLD_INVOICE_DATA field.

How Invoices Are Generated

The **pin_inv_accts** utility runs as part of your daily billing to create a regular invoice for each account that is billed on that day. Additionally, you run **pin_inv_accts** to generate corrective invoices for any corrective bills generated that day.

The **pin_inv_accts** utility runs twice to create these invoices: in the first run, invoices are generated for nonpaying bill units in a hierarchy; in the second run, invoices are generated for paying parent bill units and all nonhierarchical bill units. In both runs, this utility calls PCM_OP_INV_MAKE_INVOICE to create an invoice. This is the first opcode that gets called to create an invoice for a designated **/bill** object.

PCM_OP_INV_MAKE_INVOICE takes the POID of the bill object and calls PCM_OP_INV_POL_SELECT to determine whether custom processing or default processing is used to retrieve the events and items for billing. If custom processing is used, control is handed to PCM_OP_INV_POL_SELECT, which returns the results after the search operations are complete.

If default processing is used, PCM_OP_INV_MAKE_INVOICE performs the following steps:

- Retrieves the invoicing threshold value in the **/config/business_params** object to determine whether invoices for nonpaying bill units should be generated or consolidated into the parent account invoice. If generated, the parent account's invoice will not contain the nonpaying bill unit's invoice data.
- Retrieves all required information from the **/bill**, **/account**, **/billinfo**, and **/payinfo** objects.
  - For regular invoices, it uses the PIN_FLD_INV_TYPE field value from the **/payinfo** object to determine whether to generate a detailed invoice or a summary invoice. If the value is **1**, a summary invoice is created. If the value is **0** or **NULL**, a detailed invoice is created.
  - For corrective invoices, the opcode uses the information in the **/payinfo** object, the PIN_INV_TYPE_OF_CORRECTIVE_INVOICE and PIN_INV_CORRECTIVE__TYPE_TO_USE constants to determine whether it must generate a summary or detailed version of **Replacement Invoice** or **Invoice Correction Letter**.

  - For each billable item in an invoice correction letter, PCM_OP_INV_MAKE_INVOICE calls PCM_OP_AR_GET_ALLOCATED_AR_ITEMS which retrieves the allocated special items and additional information to filter out billable items.
  - It uses the PIN_FLD_AR_HIERARCHY_SIZE value to determine if multithreaded processing should be used. This value defines the number of nonpaying child bill units allowed for paying parent bill units. If the PIN_FLD_AR_HIERARCHY_SIZE value for a bill exceeds the invoicing threshold, and the bill is for a paying parent bill unit, PCM_OP_INV_MAKE_INVOICE retrieves the nonpaying bill units in that hierarchy by using multiple threads and processes them first.

> **Note:**
>
> The **-detail** and **-summary** parameters of **pin_inv_accts** override the PIN_FLD_INV_TYPE field value in the **/payinfo** object.

- Retrieves details about promotions purchased by the account, depending on the value of the **PromotionDetailDisplay** business parameter.
- Recalculates the event's total amounts (PIN_FLD_TOTAL) including any sponsored charges.
- Retrieves the currency information.
- If the invoice data comes from a custom application, reads the **/config/invoice_data_map** object to determine how to parse the invoice data into an flist.
- Updates the PIN_FLD_INV_FLAGS value in the **/invoice** object, which determines the type of invoice to generate.
- Creates the formatted invoice and stores it in the database.
- Calls PCM_OP_INV_POL_POST_MAKE_INVOICE to handle errors that occurred in custom operations performed by PCM_OP_INV_POL_SELECT.

When the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_INV_MAKE_INVOICE returns the invoice object but does not create the invoice object in the database. When the flag is *not* set, it returns the POID of the invoice object in the database.

> **Note:**
>
> Bill Now and on-purchase billing do not generate invoices for nonpaying bill units even when the threshold value is exceeded. The consolidated invoice for the parent (paying) bill unit is always generated.

Customizing Invoice Search Operations

You can customize the search templates during the invoicing operation so not all search operations are performed. To omit any of the search operations, or to modify the search for specific events and items, you use PCM_OP_INV_POL_SELECT, which is called by PCM_OP_INV_MAKE_INVOICE when generating invoices.

PCM_OP_INV_POL_SELECT enables you to create custom search templates to select which bill items and events are retrieved for invoices. This improves performance during invoicing

because BRM performs only selective searches and does not have to process all accounts and events in the defined invoicing period.

BRM uses the PIN_FLD_BOOLEAN field in the output flist of PCM_OP_INV_MAKE_INVOICE to determine whether the invoices should be generated using the default item and event processing in PCM_OP_INV_MAKE_INVOICE or the custom item and event processing defined in PCM_OP_INV_POL_SELECT.

- PIN_BOOLEAN_FALSE indicates PCM_OP_INV_POL_SELECT is ignored and the output flist to PCM_OP_INV_MAKE_INVOICE is the same as the flist PCM_OP_INV_POL_SELECT received as input. This is the default.
- PIN_BOOLEAN_TRUE indicates PCM_OP_INV_POL_SELECT performs the processing. The output flist to PCM_OP_INV_MAKE_INVOICE contains the input flist and the results arrays generated by the custom processing. The output flist generally contains an array of items specified for the invoice, an array of events pertaining to each item returned, and an array of information retrieved from the event's invoice data cache.

PCM_OP_INV_MAKE_INVOICE uses the return flist from PCM_OP_INV_POL_SELECT to format the invoice and store it in the database. If PCM_OP_INV_POL_SELECT was not used for processing, it returns the input flist without changes.

If errors occur during the invoicing operations, use PCM_OP_INV_POL_POST_MAKE_INVOICE to handle them. For example, you can customize this opcode to return the event that caused the invoice failure. When PCM_OP_INV_MAKE_INVOICE encounters errors returned by PCM_OP_INV_POL_SELECT, it sends the error buffer as a string to PCM_OP_INV_POL_POST_MAKE_INVOICE.

Creating Custom Search Templates

To create a custom search template for invoicing, use PCM_OP_INV_POL_SELECT. BRM uses the PIN_FLD_BOOLEAN value in the PCM_OP_INV_POL_SELECT output flist to determine whether to use the default search functionality in PCM_OP_INV_MAKE_INVOICE or the customized search functionality in this opcode.

If you customize PCM_OP_INV_POL_SELECT to search for custom event data, you must call PCM_OP_INV_DECODE_INVOICE_DATA. See "Decoding Cached Event Data for Invoicing".

To customize the invoicing search operations, set up your opcode with the following information:

1. Specify the invoice template to use.

> **Note:**
>
> The invoice templates and style sheets that define the appearance of invoices are defined in the **/config/invoice_templates** object. You can define a different appearance for invoices. To load custom style sheets, run the **pin_load_invoice_template** utility.

2. Copy the input flist to the output flist.
3. Define the search criteria for the search template. For example, create the conditions under which A/R items and events are retrieved.
4. Search for the items defined in the search template.
5. Compile the results flist.

6. Append the retrieved items to the output flist.
7. Search for events corresponding to the retrieved items.
8. Append the retrieved events to the output flist.
9. Call PCM_OP_INV_DECODE_INVOICE_DATA to decode the PIN_FLD_ INVOICE_DATA value of the **/event** object.
10. Send the result back to PCM_OP_INV_MAKE_INVOICE.

> **Note:**
>
> The events that are recorded on an invoice are defined in the **/config/invoice_events** object. By default, this object includes A/R events, such as payments and refunds, as well as cycle and usage events. You can create a **/config/invoice_events** object for custom events so that the search operation retrieves them and the specified attributes get displayed on the invoice. To specify the events recorded on an invoice, run the **pin_load_invoice_events** utility.

Example: Generating Invoices Based on Event Types

You can customize your database so that invoices contain a specific set of data based only on events you want. For example, you can generate invoices for your system that contain only adjustment and dispute information.

To do this:

1. In PCM_OP_INV_POL_SELECT, create the following search templates. In each template, customize the opcode to retrieve the items and events listed.

   Template 1

   - **/item/adjustment**
   - **/event/billing/adjustment/event**
   - **/event/billing/adjustment/item**
   - **/event/billing/adjustment/tax_event**

   Template 2

   - **/item/dispute**
   - **/event/billing/dispute/event**
   - **/event/billing/dispute/item**
   - **/event/billing/dispute/tax_event**
2. Append the search results from Template 1 and Template 2 to the output flist.
3. Customize the invoicing policy opcodes to display and print the A/R information retrieved by the search templates.

> **Note:**
>
> To generate invoices on a per-bill-unit basis, rather than a system-wide basis, you must extend the **/payinfo** storable class to define the account-specific attributes. Then customize PCM_OP_INV_POL_SELECT to perform the searches based on the flag values.

Configuring Error Checking for Customized Invoicing

To perform error checking for customized invoicing, use PCM_OP_INV_POL_POST_MAKE_INVOICE. This opcode is called after the invoice *commit* transaction to troubleshoot the errors detected by PCM_OP_INV_MAKE_INVOICE. If the **/invoice** object POID value in the PCM_OP_INV_POL_POST_MAKE_INVOICE input flist is *type* only, the invoice object was not created because errors occurred.

The PIN_FLD_ERROR_INFO array in the input and output flists contain information about the error.

You can configure PCM_OP_INV_POL_POST_MAKE_INVOICE to capture the event that caused the error and return it in the output flist.

Generating Trial Invoices

PCM_OP_BILL_MAKE_TRIAL_BILL calls PCM_OP_INV_MAKE_INVOICE to create the trial invoice. This opcode stores the trial invoice in the primary database schema as an **/invoice** object (POID type **/invoice/trial**) and returns an array of trial invoice POIDs for the invoices that were created for the account specified in the input flist. PCM_OP_BILL_MAKE_TRIAL_BILL opens a separate transaction to create the **/invoice/trial** objects.

If the PIN_FLD_PREINVOICE_MODE field is present in the input flist and has a value of **1**, PCM_OP_BILL_MAKE_TRIAL_BILL does not call the invoicing opcode to create trial invoices and only revenue assurance data is generated for the account. If the PIN_FLD_PREINVOICE_MODE field is not present in the input flist or has a value of **0**, trial billing creates trial invoices.

> **Note:**
>
> If a start date is not provided in the input flist, PCM_OP_BILL_MAKE_TRIAL_BILL performs trial billing for all billing cycles completed before the end date. For accounts with skipped billing cycles, more than one trial invoice might be created.

How Trial Invoicing Works for Hierarchical Bill Units in Oracle Analytics Publisher

You generate a trial invoice to validate billing charges and check for inaccuracies before creating the final bill.

When you run the **pin_trial_bill_accts** utility to perform trial billing with Oracle Analytics Publisher, PCM_OP_BILL_MAKE_TRIAL_BILL is called to create the trial invoices and collect revenue assurance data for the trial billing run.

During trial invoicing for hierarchical bill units, PCM_OP_ BILL_MAKE_TRIAL_BILL passes the PIN_FLD_INV_FLAGS field's value to this opcode. This opcode decides from the value whether to create a nonpaying child or parent trial invoice:

- If the value is PIN_INV_TYPE_SUBORDINATE, a nonpaying child invoice is created.
- If the value is PIN_INV_TYPE_PARENT, a parent invoice is created.

In hierarchical bill unit billing, charges for the nonpaying child bill units are calculated first and then rolled up to the parent bill unit as follows:

1. When the **pin_trial_bill_accts** utility is run with the **-pay_type** parameter and the payment method subordinate ID **10007**, PCM_OP_BILL_MAKE_TRIAL_BILL creates a **/invoice/trial** object for each nonpaying child bill unit.
2. When the **pin_trial_bill_accts** utility is run with the **-pay_type** parameter and the specified payment method parent ID, PCM_OP_BILL_MAKE_TRIAL_BILL is called to trial bill each bill unit. PCM_OP_BILL_MAKE_TRIAL_BILL calculates the parent bill unit bill totals using the nonpaying child totals from PCM_OP_BILL_MAKE_BILL and generates parent invoices.
3. PCM_OP_BILL_MAKE_TRIAL_BILL passes the PIN_FLD_INV_FLAGS field's value to PCM_OP_INV_MAKE_INVOICE. PCM_OP_INV_MAKE_INVOICE decides from the value whether to create a nonpaying child or paying parent trial invoice:
   - If the value is PIN_INV_TYPE_SUBORDINATE, a nonpaying child invoice is created.
   - If the value is PIN_INV_TYPE_PARENT, a paying parent invoice is created.
4. PCM_OP_INV_MAKE_INVOICE calls PCM_OP_INV_POL_PREP_INVOICE, which prepares invoice information prior to formatting and storing.

   PCM_OP_INV_POL_PREP_INVOICE checks the value of the PIN_FLD_FLAGS field. If the value is PIN_INV_TYPE_TRIAL_INVOICE plus PIN_INV_TYPE_SUBORDINATE, PCM_OP_INV_POL_PREP_INVOICE adds the PIN_FLD_BILLS array and the PIN_FLD_CHECK_SPLIT_FLAGS field, which are required to generate the trial invoices for the nonpaying child's parent.

   The PIN_FLD_BILLS array contains the following fields:

   - PIN_FLD_START_T
   - PIN_FLD_END_T
   - PIN_FLD_AR_BILLINFO_OBJ
   - PIN_FLD_DUE
   - PIN_FLD_ADJUSTED
   - PIN_FLD_WRITEOFF
   - PIN_FLD_DISPUTED
   - PIN_FLD_RECVD

The PIN_FLD_CHECK_SPLIT_FLAGS is used internally by Oracle Analytics Publisher to collate nonpaying child invoices into the final invoice.

About Associating Bill Units with an Oracle Analytics Publisher Invoice and Report

When the BRM-Oracle Analytics Publisher integration is enabled, BRM automatically associates bill units with a Oracle Analytics Publisher report name and invoice template name during the account creation process. When an account is created, BRM performs the following for each bill unit in the account:

- Determines the Oracle Analytics Publisher invoice report and template that the bill unit qualifies for by reading the **/config/business_profile** object. This object defines your invoice types, the criteria a bill unit must meet for the invoice type, and the Oracle Analytics Publisher invoice report and template names associated with the invoice type. For example, a **/config/business_profile** might define the following in Table 15-5 for a regular bill:

Table 15-5 Business Profile Example

Invoice Type	Criteria	Oracle Analytics Publisher invoice Report and Template Names
Monthly billing	The **/billinfo** object's PIN_FLD_WHEN field is set to **1**.	**monthly_invoice_report** and **monthly_invoice**
Quarterly billing	The **/billinfo** object's PIN_FLD_WHEN field is set to **3**.	**quarter_invoice_report** and **quarter_invoice**

- Creates an **/associated_bus_profile** object for the bill unit. This object stores the template and report names, and type of invoice associated with invoicing for the bill unit, as an element in a template array called PIN_FLD_TEMPLATE_ARRAY.
  - PIN_FLD_TEMPLATE_NAME. The Template name configured in the pair key Template_Name of the corresponding invoicing **/config/business_profile** object.
  - PIN_FLD_REPORT_NAME. The Report name configured in the pair key Report_Name of the corresponding invoicing **/config/business_profile** object.
  - PIN_INV_TYPE. The type of invoice stored as PIN_INV_TYPE_REGULAR or PIN_INV_TYPE_CORRECTIVE.
- Associates the bill unit with its **/associated_bus_profile** object by populating the **/billinfo** object's PIN_FLD_ASSOC_BUS_PROFILE_OBJ_LIST field.

BRM creates an **/associated_bus_profile** object for each bill unit in an account by calling the PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE opcode. This opcode is called internally by the PCM_OP_CUST_CREATE_BILLINFO opcode during the account creation process.

Whenever invoice business profiles are modified in the **/config/business_profile** object, use the PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE opcode to update all related **/associated_bus_profile** objects.

About the /associated_bus_profile Object

If you are using BRM-Oracle Analytics Publisher integration framework to generate invoice documents, at the time of customer account creation, the **/associated_bus_profile** object is created. The **/associated_bus_profile** object stores the invoicing business profile information

for a **/billinfo** object. The **/associated_bus_profile** object contains the layout template name in the PIN_FLD_TEMPLATE _NAME field and the report name in the PIN_FLD_REPORT_NAME field.

Creating /associated_bus_profile Objects

If the BRM-Oracle Analytics Publisher invoicing integration is enabled, during customer account creation, internally the PCM_OP_CUST_CREATE_BILLINFO opcode calls the PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE opcode to create one **/associated_bus_profile** object for each bill unit in the account.

16

Job Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) job opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Running BRM Applications
- Running Business Operation Jobs
- Managing Business Operations Job Templates
- Managing Business Operations Jobs

Opcodes Described in This Chapter

Table 16-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 16-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_JOB_EXECUTE	Running BRM Applications Running Business Operation Jobs
PCM_OP_JOB_PROCESS_TEMPLATE	Managing Business Operations Job Templates
PCM_OP_JOB_PROCESS_DEFINITION	Creating Custom Jobs Modifying Custom Jobs Deleting Custom Jobs
PCM_OP_JOB_POL_VALIDATE_DEFINITION	Validating Job Definitions
PCM_OP_JOB_POL_POST_PROCESS_DEFINITION	Post-Processing Job Definitions

Running BRM Applications

> **Note:**
>
> Before calling this opcode, ensure that:
>
> - The **pin_job_executor** process is listening for calls from PCM_OP_JOB_EXECUTE. See "Configuring pin_job_executor to Listen for Opcode Calls" in *BRM System Administrator's Guide*.
> - The opcode is configured to run BRM applications. See "Configuring PCM_OP_JOB_EXECUTE to Run BRM Applications" in *BRM System Administrator's Guide*.
> - The **pin_job_executor** process is running in the background. To start the process, run this command from the **$PIN_HOME/bin** directory:
>
> ```
> ./start_pje
> ```

To run **pin_virtual_time** or another BRM utility, use the PCM_OP_JOB_EXECUTE opcode and pass in the following data:

- The POID of the account calling the opcode.
- PIN_FLD_FLAG set to **0**, which indicates to run the application in a non-job context.
- The utility command to run, such as **my_custom_utility -f filename -v**.
- The working directory, relative to **$PIN_HOME**, in which to run the utility command. For example, for a working directory of $PIN_HOME/sys/cm, you would enter **sys/cm**.

> **Note:**
>
> Enter an absolute path to the working directory. Do not include **/../** or **./** in the path.

PCM_OP_JOB_EXECUTE then performs these operations:

1. Opens a non-job context.
2. Creates an **/event/activity/job_request** object for auditing purposes.
3. Validates that the specified utility is located under the **$PIN_HOME** directory.
4. Validates that the specified path to the working directory is an absolute path.
5. Validates that the working directory specified in the PIN_FLD_DIRECTORY input flist field matches one of the allowed directories in the **pin_job_executor** utility's **Infranet.properties** file.
6. Runs the utility commands passed in the opcode's input flist.
7. Returns the result of the utility's execution in the opcode's output flist.

The following shows a sample PCM_OP_JOB_EXECUTE input flist for running a BRM application in a non-job context:

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_FLAGS            INT [0] 0
0 PIN_FLD_INSTRUCTION      STR [0] "pin_virtual_time -m 2 01010000"
0 PIN_FLD_DIRECTORY        STR [0] "apps/pin_billd"
```

In this example, PCM_OP_JOB_EXECUTE would run the following command from the **$PIN_HOME/apps/pin_billd** directory:

```
pin_virtual_time -m 2 01010000
```

Running Business Operation Jobs

> **Note:**
>
> Before calling this opcode, ensure that:
>
> - The **pin_job_executor** process is listening for calls from PCM_OP_JOB_EXECUTE. See "Configuring pin_job_executor to Listen for Opcode Calls" in *BRM System Administrator's Guide*.
> - The **pin_job_executor** process is running in the background. To start the process, run this command from the *BRM_home***/bin** directory:
>
>   **./start_pje**

To run business operations jobs, such as billing, invoicing, and payment collections, use the PCM_OP_JOB_EXECUTE opcode and pass in the following data:

- The POID of the account calling the opcode.
- PIN_FLD_FLAG set to **1**, which indicates to run applications within a job context.
- The parameters and values to run, provided as a set of key-value pairs in the PIN_FLD_ARGS array. Table 16-2 lists the supported parameters.

Table 16-2 PCM_OP_JOB_EXECUTE Parameters for Business Operations Jobs

Parameter	Description	Mandatory
job_template	Specifies the ID of the job template to run. The job template contains information about the utility that is run as well as the database schema numbers.	Yes
request_id	Specifies the job request ID to use.	Yes
start	Specifies the start time of the job. The default is immediately.	No
end	Specifies the end time of the job.	Yes
job_id	Re-runs the job with the specified job ID.	No

This opcode is called by Business Operations Center.

PCM_OP_JOB_EXECUTE then performs these operations:

1. Opens a job context.
2. Calls the **pin_job_executor** utility with the parameters passed in the PIN_FLD_ARGS array.
3. The **pin_job_executor** utility does the following:
   a. Extracts the database schema numbers from the specified **/job_template** object.
   b. Creates a **/job/boc** object for each database schema.
   c. For each **/job/boc** object, a separate thread runs the business operations utility specified in the **/job_template** object:
      - For billing jobs, it runs the **pin_bill_accts** utility.
      - For payment collections jobs, it runs the **pin_collect** utility.
      - For invoicing jobs, it runs the **pin_inv_accts** utility.
      - For general ledger report jobs, it runs the **pin_ledger_report** utility.
      - For synchronizing pricing data, it runs the **pin_export_price** utility.
      - For refund jobs, it runs the **pin_refund** utility.
      - For custom jobs, it runs the utility that you configure.
   d. For each **/job/boc** object, generates an **/event/notification/job_complete** event.
4. Returns the result of the job and the POID of the **/job/boc** object in the opcode's output flist.

The following shows a sample PCM_OP_JOB_EXECUTE input flist:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_FLAGS              INT [0] 1
0 PIN_FLD_ARGS             ARRAY [0] allocated 2, used 2
1     PIN_FLD_NAME           STR [0] "job_template"
1     PIN_FLD_VALUE          STR [0] "119188"
0 PIN_FLD_ARGS             ARRAY [1] allocated 2, used 2
1     PIN_FLD_NAME           STR [0] "request_id"
1     PIN_FLD_VALUE          STR [0] "pje_js_1"
0 PIN_FLD_ARGS             ARRAY [2] allocated 2, used 2
1     PIN_FLD_NAME           STR [0] "end"
1     PIN_FLD_VALUE          STR [0] "10/08/2021"
```

Managing Business Operations Job Templates

To create, modify, or delete a business operations job template, use the PCM_OP_JOB_PROCESS_TEMPLATE opcode and pass in the following data:

- The POID of the **/job_template** object.
- The POID of the **/account** object that is calling the opcode.
- The database schema on which to run the business operations job: **0** specifies all schemas, **1** specifies the primary schema, **2** specifies the second schema, and so on.
- The name of the program that is calling the opcode.
- The type of business operations job to run such as billing, collections, or invoicing.
- The action to perform on the job template: create, modify, or delete.

- The name of the Business Operations Center user.
- Other optional configuration information.

When called, PCM_OP_JOB_PROCESS_TEMPLATE does the following:

- Validates the database schema number passed in the input flist and then creates, modifies, or deletes the specified **/job_template** object. By default, this opcode is called by Business Operations Center.
- Throws an error on the **Create** action if the **/job_definition** is marked with the status **Closed**.
- Indicates whether the **/job_template** object was deleted or marked as closed.

Managing Business Operations Jobs

You can manage your custom Business Operations jobs by creating, modifying, or deleting job definition objects. Additionally, you can validate and carry on the post-processing of the job definitions.

- Creating Custom Jobs
- Modifying Custom Jobs
- Deleting Custom Jobs
- Validating Job Definitions
- Post-Processing Job Definitions

Creating Custom Jobs

You create jobs using the PCM_OP_JOB_PROCESS_DEFINITION opcode, which creates **/job_definition** objects. This opcode is called by Business Operations Center.

To create custom business operations jobs, use the PCM_OP_JOB_PROCESS_DEFINITION opcode and pass in the following data:

- The POID of the job definition to be created
- The user and account information that is calling the job
- The definition of the job, including the category and the resource name
- The action to be performed: in this case **Create**

PCM_OP_JOB_PROCESS_DEFINITION then performs these operations for the **Create** action:

- Validates that all mandatory fields for creating a job definition are provided in the input flist and that the resource name does not match any existing resource name (for example, the predefined names such as Billing or Invoicing).
- Calls PCM_OP_JOB_POL_VALIDATE_DEFINITION to perform further validation of the job definition.
- If all the validation checks are successful, calls PCM_OP_CREATE_OBJ to create the job definition object.
- Calls the policy opcode PCM_OP_JOB_POL_POST_PROCESS_DEFINITION with the return flist, for additional post-processing.
- Returns the PIN_FLD_POID of the created job definition or sets an error if the PCM_OP_CREATE_OBJ opcode fails.

Modifying Custom Jobs

You modify jobs using the PCM_OP_JOB_PROCESS_DEFINITION opcode, which modifies **/job_definition** objects. This opcode is called by Business Operations Center.

To modify custom business operations jobs, use the PCM_OP_JOB_PROCESS_DEFINITION opcode and pass in the following data:

- The POID of the job definition to be modified
- The user and account information that is calling the job
- The fields that should be modified for the job, including the category
- The action to be performed: in this case **Modify**

PCM_OP_JOB_PROCESS_DEFINITION then performs these operations for the **Modify** action:

- Validates that the resource name is not provided. If it is provided, it generates an error. The resource name cannot be modified.
- Calls PCM_OP_WRITE_FLDS to update the job definition object.
- Calls the policy opcode PCM_OP_JOB_POL_POST_PROCESS_DEFINITION with the return flist, for additional post-processing.
- Returns the PIN_FLD_POID of the modified job definition or sets an error if the PCM_OP_WRITE_FLDS opcode fails.

Deleting Custom Jobs

You delete jobs using the PCM_OP_JOB_PROCESS_DEFINITION opcode, which deletes **/job_definition** objects. This opcode is called by Business Operations Center.

To delete custom business operations jobs, use the PCM_OP_JOB_PROCESS_DEFINITION opcode and pass in the following data:

- The POID of the job definition to be deleted
- The user and account information that is calling the job
- The name of the program or application from which this opcode is called
- The action to be performed: in this case **Delete**

PCM_OP_JOB_PROCESS_DEFINITION then performs these operations for the delete action:

- Validates that the provided POID exists.
- Calls the PCM_OP_JOB_POL_VALIDATE_DEFINITION policy opcode for further validation and to ensure that the job definition can be deleted.
- Checks if the **/job_definition** object has any **/job_template/custom** instances associated with it to ensure that the job definition is not being used by any pending jobs.
- If the job definition is being used by any pending jobs, it is not deleted.
- Otherwise, the opcode deletes the **/job_definition** object using PCM_OP_DELETE_OBJ.
- Creates an event **(/event/job_definition/delete)** to log the deletion of the job definition.
- Searches for and deletes any associated **/job_template/custom** objects by calling PCM_OP_JOB_PROCESS_TEMPLATE with the **Delete** action on each result.

- Calls PCM_OP_JOB_POL_POST_PROCESS_DEFINITION with the return flist.
- Returns the result of PCM_OP_JOB_POL_POST_PROCESS_DEFINITION if the process is successful; otherwise, sets an error.

Validating Job Definitions

To validate the contents of a job definition, use the PCM_OP_JOB_POL_VALIDATE_DEFINITION opcode. This policy opcode is called by the PCM_OP_JOB_PROCESS_DEFINITION opcode during creation and modification.

To use the PCM_OP_JOB_POL_VALIDATE_DEFINITION opcode, pass in the following data:

- The POID of the job definition
- Name of the program from where this opcode is called
- Name of the BOC user making the call
- Action to be performed on the **/job_definition** object. It can be **Create**, **Modify**, or **Delete**
- Job Resource name: It should be unique

PCM_OP_JOB_POL_VALIDATE_DEFINITION then performs the following tasks:

- Validates the provided information for a job definition object, ensuring that all required and relevant fields are correctly populated.
- Checks if essential parameters like PIN_FLD_NAME, PIN_FLD_LOCATION, PIN_FLD_PROGRAM_NAME, PIN_FLD_USER_NAME, and PIN_FLD_RESOURCE_NAME are provided and within valid limits.
- Validates the action specified in PIN_FLD_ACTION is either **Create**, **Modify**, or **Delete**.
- If provided, it validates the job category in PIN_FLD_CATEGORY. The category should be unique, and it should not match any predefined category names, especially when creating a job definition.
- Validates the job definition's status in PIN_FLD_STATUS. It should be set to **active** or **closed**. If not provided, the default status is **active**.
- Ensures that the provided PIN_FLD_ACCOUNT_OBJ is associated with the job definition.
- The opcode returns the validated job definition POID for further processing.

Post-Processing Job Definitions

You can perform post-processing validation and actions on job definition objects and related templates with PCM_OP_JOB_POL_POST_PROCESS_DEFINITION opcode. It is typically used after specific operations, such as deletion or closure of job definitions, to ensure that any dependent templates are handled correctly and that the overall integrity of the job definition data is maintained.

To use the PCM_OP_JOB_POL_POST_PROCESS_DEFINITION opcode, pass in the following data:

- The POID of the job definition

PCM_OP_JOB_POL_POST_PROCESS_DEFINITION then performs the following tasks:

- Checks the PIN_FLD_DELETED_FLAG when the action is "Delete" to determine whether the **job_definition** was deleted or marked as closed.

- Processes the PIN_FLD_TEMPLATE_ARRAY, an array that may hold schema numbers and associated job templates.
- For each template in the array, checks the PIN_FLD_DELETED_FLAG to determine if the **job_template** was deleted or marked as closed.
- Updates the job definition's status or attributes based on the performed action.
- Returns the updated job definition's POID after post-processing.

17

Loan Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) loan opcode workflows.

For information about the loan opcodes, see "Loan FM Standard Opcodes" and "Loan FM Policy Opcodes".

Topics in this document:

- Opcodes Described in This Chapter
- Granting a Loan
- Granting a Channel Loan
- Resetting a Loan Cycle and Other Loan Profile Fields
- Checking a Customer's Eligibility for a Loan
- Retrieving Loan Information
- Creating and Modifying a Customer's Loan Profile
- Customizing Loan Recovery

Opcodes Described in This Chapter

Table 17-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 17-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_CREATE_PROFILE	Creating and Modifying a Customer's Loan Profile
PCM_OP_CUST_MODIFY_PROFILE	Creating and Modifying a Customer's Loan Profile
PCM_OP_LOAN_APPLY_LOAN	Granting a Loan
PCM_OP_LOAN_ELIGIBILITY	Checking a Customer's Eligibility for a Loan
PCM_OP_LOAN_GET_LOAN	Retrieving Loan Information
PCM_OP_LOAN_POL_ELIGIBILITY	Customizing Loan Eligibility Checks
PCM_OP_LOAN_POL_PRE_APPLY_LOAN	Customizing Loan Grants
PCM_OP_LOAN_POL_PRE_RECOVER_LOAN	Customizing Loan Recovery
PCM_OP_LOAN_POL_RESET_CYCLE	Resetting a Loan Cycle and Other Loan Profile Fields

Table 17-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_RATE_POL_POST_RATING	Granting a Channel Loan

Granting a Loan

To grant a loan to a customer, use PCM_OP_LOAN_APPLY_LOAN. This opcode is not called internally by BRM, you must configure API calls from external systems to call this opcode.

This opcode does the following:

1. Takes at minimum an account POID as input. You can optionally provide locale, amount, and service parameters. If the account POID alone is provided, rather than the full account object, the account object is retrieved using the service object or service type and MSISDN.
2. Calls the PCM_OP_CUST_FIND_PROFILE opcode to retrieve the customer's loan profile.
3. If PIN_FLD_UNIT and PIN_FLD_FREQUENCY are specified in the **/profile/loan** object and the current date is later than the timestamp in PIN_FLD_CYCLE_T, calls the PCM_OP_LOAN_POL_RESET_CYCLE opcode to reset the loan cycle.
4. Calls the PCM_OP_LOAN_ELIGIBILITY opcode to check if the customer is eligible for the requested loan.
   If the customer is ineligible, the loan is rejected and a response containing the rejection reason is returned to the calling service.

   If the customer is eligible, the workflow continues.
5. If there is a fee configured in the **/config/loan** object specified in the **/profile/loan**, calls PCM_OP_ACT_USAGE to create the **/event/billing/loan_fee** event and **/item/loan_fee** as a non-billable open item.
6. Calls PCM_OP_ACT_USAGE to create **/event/billing/loan_credit** and **/event/billing/loan_debit** events and **/item/loan_credit** and **/item/loan_debit** items.
   The credit represents the cash granted and the debit represents the loan asset in a form of double-entry bookkeeping. This allows the customer's account balance to remain at 0 while their credit limit is temporarily increased by the loan amount.

   The **gl_id** and **gl_accts** fields are updated in the **reasons.en_US** and **pin_glid** files for the events.
7. If the input flist contains a noncurrency resource specified in the PIN_FLD_RESOURCE_ID field, creates an **/event/billing/loan_grant** event to grant the noncurrency resource configured for the event in PDC or Pricing Center.
8. Calls the PCM_OP_BILL_SET_LIMIT_AND_CR opcode to set the amount owed for the loan in the balance group in the PIN_FLD_OUSTANDING_AMOUNT field to the sum of:
   - The loan amount
   - The loan service fee
   - The tax on the loan service fee
   - The tax on the loan amount
9. Calls PCM_OP_CUST_MODIFY_PROFILE to update the loan profile to reflect the total loan amount and the number of loans granted.

10. The output flist containing information about the loan granted, including amount, resource ID, and events, is returned to the calling service.

Customizing Loan Grants

To customize loan grants, use the PCM_OP_LOAN_POL_PRE_APPLY_LOAN policy opcode. By default, this policy opcode does nothing.

This policy opcode is called by PCM_OP_LOAN_APPLY_LOAN.

Granting a Channel Loan

To grant a channel loan for a package without considering or consuming a customer's available balance, customize the PCM_OP_RATE_POL_POST_RATING policy opcode to call PCM_OP_LOAN_APPLY_LOAN and grant a loan for the price of the package purchased by subscription request from your channel.

For example, you can indicate that the package should be purchased using a loan with the PIN_FLD_FLAGS field in the input flist for PCM_OP_RATE_POL_POST_RATING, then use PCM_OP_RATE_POL_POST_RATING to get the package's rated impact for the loan amount.

Resetting a Loan Cycle and Other Loan Profile Fields

You use loan cycles when limiting how many times a loan can be granted and total amounts that can be granted.

To reset a loan cycle, use PCM_OP_LOAN_POL_RESET_CYCLE. This opcode is called by the PCM_OP_LOAN_APPLY_LOAN opcode.

This opcode does the following:

1. Takes at minimum an account POID as input. You can optionally provide a service object as well.
2. Calls the PCM_OP_CUST_FIND_PROFILE opcode to retrieve the customer's loan profile.
3. Uses the offset and unit from the PCM_OP_CUST_FIND_PROFILE output flist to calculate the next loan cycle date.
4. Validates that the current date is later than the timestamp in PIN_FLD_CYCLE_T in the **/profile/loan** object retrieved.
5. Calls the PCM_OP_CUST_MODIFY_PROFILE opcode to set the following fields in the **/profile/loan** object:
   - PIN_FLD_CREDIT_AMOUNT: Resets to 0
   - PIN_FLD_NUMBER_OF_LOANS: Resets to 0
   - PIN_FLD_CYCLE_T: Sets to newly calculated date

You can also customize this policy opcode to reset other fields in the **/profile/loan** object.

Checking a Customer's Eligibility for a Loan

To check whether a customer is eligible for a loan, use PCM_OP_LOAN_ELIGIBILITY. This opcode is called by PCM_OP_LOAN_APPLY_LOAN.

This opcode does the following:

1. Takes at minimum an account POID as input. You can optionally provide locale, amount, and service parameters.
2. Determines the account's location in one of the following ways:
   - Using the value provided in the LOCALE field
   - Looking the location up using the ZONEMAP_TARGET and ZONEMAP_NAME fields. If ZONEMAP_NAME isn't also specified, the name is taken from the **DefaultZoneMapName** Subscription business parameter.
   - If neither locale nor zonemap details are included in the input flist, the city, state, and zipcode specified in the customer's PIN_FLD_NAMEINFO array are used.
3. Determines if the account is eligible by confirming that all of the following are true:
   - The sum of PIN_FLD_CREDIT_AMOUNT in **/profile/loan** and the loan amount requested doesn't exceed the fixed or scaled maximum configured in **/config/loan**.
   - The number in PIN_FLD_NUMBER_OF_LOANS in **/profile/loan** isn't already equal to PIN_FLD_QUANTITY in **/config/loan**.
   - The account's location is eligible, depending on the configuration in PIN_FLD_LOCATION_MODE and the LOCATIONS array in **/config/loan**.
   - The amount requested is greater than PIN_FLD_LOAN_MINIMUM in **/config/loan**.
   - The account's age is greater than the amount specified in PIN_FLD_AGE and PIN_FLD_UNIT in **/config/loan**.
   - The account's status is active.
4. Calls the PCM_OP_LOAN_POL_ELIGIBILITY policy opcode.
5. Returns a response that includes one of the reason codes in Table 17-2.

Table 17-2 Reason Codes

Code	Description
0	Success. The account is eligible for the loan.
1	Failure. The maximum loan amount has been exceeded.
2	Failure. A loan has already been granted the maximum number of times.
3	Failure. The account's location is not eligible for the loan.
4	Failure. An external entity (called by the policy opcode) rejected the loan request.
5	Failure. The loan amount is less than the configured minimum.
6	Failure. The account has not been active long enough.
7	Failure. The account's status is inactive.
8	Failure. Another error occurred, such as missing or invalid information.
9	Failure. The customer has opted out of loans.
10	Failure. The customer has met or exceeded the maximum number of active loans.

6. Generates one of the following events:
   - **/event/notification/loan_reject**
   - **/event/notification/loan_success**

Customizing Loan Eligibility Checks

To customize loan eligibility checks before they are finalized, use the PCM_OP_LOAN_POL_ELIGIBILITY policy opcode. By default, this policy opcode does nothing.

This policy opcode is called by PCM_OP_LOAN_ELIGIBILITY.

Retrieving Loan Information

To retrieve loan information, use PCM_OP_LOAN_GET_LOAN. This opcode is not called internally by BRM, you must configure API calls from external systems to call this opcode.

This opcode takes an account POID as input. You can optionally set the value of PIN_FLD_FLAGS to **0** to return only active loans or **1** to return all loans, active and past.

The output flist contains

- Event arrays for each loan that list:
  - The POIDs of the **/event/billing/loan_debit** event, **/event/billing/loan/fee** event, and associated service
  - The total outstanding amount that has yet to be paid for that loan
  - The amount originally loaned
  - The loan fee
  - The loan tax
  - The loan type
  - The loan channel
- The maximum loan amount configured in the **/profile/loan** object
- The amount available to be loaned, calculated based on the maximum configured in the **/profile/loan** object or the **/config/loan** object and the current total outstanding amount for all active loans. If the outstanding amount is greater than the maximum, the amount available is set to **0**.

Creating and Modifying a Customer's Loan Profile

You create a loan profile for customers to connect the account or service to a loan configuration. When the customer is granted a loan, the information for that cycle, such as amount loaned and number of loans, is stored in the loan profile.

To create a loan profile, use PCM_OP_CUST_CREATE_PROFILE. This opcode is called by PCM_OP_CUST_COMMIT_CUSTOMER. To modify the fields in a loan profile, use PCM_OP_CUST_MODIFY_PROFILE. See "Managing and Customizing Profiles" and "Creating Accounts" for more information about using these opcodes generally.

The input flist for creating a loan profile requires, at minimum, an account object, profile name (which corresponds to the name of the **/config/loan** object you are connecting the profile to), the maximum number of active loans, eligibility, and loan cycle frequency. You can optionally specify:

- Service details

- Thresholds for offering loans (these are stored in the credit profile of the associated balance group)
- Loan repayment details
- Maximum loan amounts (which overrides the fixed or scaled maximum amount specified in the **/config/loan** object)

For example, to create a profile:

```
0 PIN_FLD_POID                      POID      [0] 0.0.0.1 /profile/loan -1 0
0 PIN_FLD_ACCOUNT_OBJ                         POID      [0] 0.0.0.0 /account -1
0
0 PIN_FLD_SERVICE_OBJ                         POID      [0] 0.0.0.0 /service/
telco/gsm -1 0
0 PIN_FLD_SERVICE_ID                          STR       [0] "0049100098"
0 PIN_FLD_PROFILES                            ARRAY     [0] allocated 20, used 7
1    PIN_FLD_PROFILE_OBJ                      POID      [0] 0.0.0.1 /profile/
loan -1 0
1    PIN_FLD_INHERITED_INFO                   SUBSTRUCT [0] allocated 20, used 5
2        PIN_FLD_LOAN_INFO                    SUBSTRUCT [0] allocated 20,used 4
3            PIN_FLD_EXTERNAL_ELIGIBILITY     INT       [0] 1
3            PIN_FLD_PROFILE_NAME             STR       [0] "Loan Configuration
1"
3            PIN_FLD_FREQUENCY                INT       [0] 30
3            PIN_FLD_UNIT                     ENUM      [0] 4
3            PIN_FLD_OPT_LOAN                 INT       [0] 1
3            PIN_FLD_MAX_ACTIVE_LOANS         INT       [0] 3
3            PIN_FLD_REPAYMENT_DAYS           INT       [0] 14
3            PIN_FLD_PULLBACK                 INT       [0] 1
3            PIN_FLD_LOAN_THRESHOLDS_FIXED    STR       [0] "|15|20"
2        PIN_FLD_LIMIT                        ARRAY     [0] allocated 20,used 4
3            PIN_FLD_VALID_FROM               TSTAMP    [0] (1659423601) Tue
Aug  2 00:00:01 2022
3            PIN_FLD_CREDIT_LIMIT             DECIMAL   [0] 100
```

This example specifies the following details:

- **Loan cycle details**: PIN_FLD_FREQUENCY and PIN_FLD_UNIT specify that the loan cycle resets every 30 days.
- **Threshold details**: PIN_FLD_LOAN_THRESHOLDS_FIXED field specifies to offer a loan when the customer's balance reaches 20, then 15. These fields can be set for packages at design time and updated for individual customers in their loan profiles.
- **Repayment details**:
  - PIN_FLD_REPAYMENT_DAYS specifies that the customer must pay the loan back in 14 days.
  - PIN_FLD_PULLBACK specifies that any outstanding balance that remains after 14 days will be pulled from the customer's available balance.
- **Maximum loan details:**
  - PIN_FLD_MAX_ACTIVE_LOANS specifies that the customer can have a maximum of 3 active loans at a time.
  - The PIN_FLD_LIMIT array specifies a maximum loan amount of 100, starting on August 2, 2022. This overrides any maximums specified in the **/config/loan** object for the associated loan.

To modify the same profile:

```
0 PIN_FLD_POID POID [0] 0.0.0.1  /profile/loan 239534 0
0 PIN_FLD_PROFILES ARRAY [0] allocated 20, used 7
1    PIN_FLD_PROFILE_OBJ POID [0] 0.0.0.1  /profile/loan 239534 0
1    PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 5
2        PIN_FLD_LOAN_INFO SUBSTRUCT [0] allocated 10, used 4
3            PIN_FLD_LOAN_THRESHOLDS INT [0] 80
2        PIN_FLD_LIMIT ARRAY [0] allocated 20,used 4
3            PIN_FLD_VALID_FROM TSTAMP [0] (1675324802)  Thu Feb  2 00:00:02
2023
3            PIN_FLD_CREDIT_LIMIT DECIMAL [0] 200
```

In this example:

- PIN_FLD_LOAN_THRESHOLDS specifies to offer a loan when 80% of the customer's balance is consumed.
- The PIN_FLD_LIMIT array adds a new maximum loan amount of 200, starting February 2, 2023.

Customizing Loan Recovery

To customize the loan recovery process, use the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN policy opcode. By default, this policy opcode does nothing.

This policy opcode is called by internal loan recovery opcodes, before transferring balances from payment or balance transfer events to loan items.

18

Notification Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) notification opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Creating Notification Specifications
- Modifying Notification Specifications
- Retrieving Notification Specifications
- Managing In-Advance and Post-Expiration Notifications
- Calculating the Notification Delivery Time
- Verifying Customer Opt-In Preferences
- Retrieving Last Notification Time and Offset
- Customizing Notifications

For more information about notification specifications, see "Sending Messages to Customers through External Notification Applications" in *BRM Managing Customers*.

Opcodes Described in This Chapter

Table 18-1 describes the notification management opcodes.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 18-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_NOTIFICATION_CALC_DELIVERY_TIME	Calculating the Notification Delivery Time
PCM_OP_NOTIFICATION_CREATE_SPECIFICATION	Creating Notification Specifications
PCM_OP_NOTIFICATION_GET_LASTNOTIFY_TSTAMP	Retrieving Last Notification Time and Offset
PCM_OP_NOTIFICATION_GET_SPECIFICATION	Retrieving Notification Specifications
PCM_OP_NOTIFICATION_MODIFY_SPECIFICATION	Modifying Notification Specifications
PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION	Customizing Notification Retrieval
PCM_OP_NOTIFICATION_POL_PREP_SPECIFICATION	Preparing Notification Specifications

Table 18-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION	Validating Notification Specifications
PCM_OP_NOTIFICATION_PROCESS_NOTIFICATION	Managing In-Advance and Post-Expiration Notifications
PCM_OP_NOTIFICATION_VERIFY_PUBLISH_REQUIRED	Verifying Customer Opt-In Preferences

Creating Notification Specifications

Use the PCM_OP_NOTIFICATION_CREATE_SPECIFICATION opcode to create notification specifications, defining the customers who can receive messages through an external notification application and how and when to deliver the messages. This opcode creates a **/config/notification_spec** object.

You set the notification specification name, associated business event, validity dates, silent period mode, notification type, aggregation mode, opt-in type, and offset value by providing the information in the input flist.

You specify to send notifications in advance or after an event occurs by including the following in the input flist:

- PIN_FLD_NOTIFICATION_SPEC.PIN_FLD_NOTIFICATION_TYPE: The type of notification set to in advance (**1**) or after an event occurs (**5**).
- PIN_FLD_ADVANCE_NOTIFICATIONS.PIN_FLD_OFFSET_VALUE: The amount of time before or after an event to send a notification, such as 5 or 10.
- PIN_FLD_ADVANCE_NOTIFICATIONS.PIN_FLD_OFFSET_UNIT: The unit for the offset value, which can be minutes (**2**), hours (**3**), days (**4**), months (**5**), years (**6**), or weeks (**10**).

Set the input flist spec under the PIN_FLD_NOTIFICATION_SPEC.PIN_FLD_AGGREGATE_MODE: Whether to send individual notifications for each triggering event (0) or aggregate all expiring candidates' details at respective search level

The PCM_OP_NOTIFICATION_CREATE_SPECIFICATION opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Retrieves the optional fields from the database and sets them in the input flist.
3. Calls the PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION policy opcode to validate the input parameters. See "Validating Notification Specifications".
4. Searches the database for an existing **/config/notification_spec** object with the same notification specification name. If one already exists, the opcode sends an error.
5. Calls the PCM_OP_NOTIFICATION_POL_PREP_SPECIFICATION policy opcode to prepare the input parameters. See "Preparing Notification Specifications".
6. Calls the PCM_OP_CREATE_OBJ opcode to create the **/config/notification_spec** object.

Modifying Notification Specifications

Use the PCM_OP_NOTIFICATION_MODIFY_SPECIFICATION opcode to modify attributes in a specified notification specification (**/config/notification_spec** object).

> **Note:**
>
> You can modify any field in the **/config/notification_spec** object except PIN_FLD_NAME and PIN_FLD_EVENT_NAME.

The PCM_OP_NOTIFICATION_MODIFY_SPECIFICATION opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION policy opcode to perform validations. See "Validating Notification Specifications".
- Searches for the **/config/notification_spec** object that matches the specified POID.
- Calls the PCM_OP_WRITE_FLDS opcode to update the **/config/notification_spec** object with the fields provided in the input flist.

Retrieving Notification Specifications

Use the PCM_OP_NOTIFICATION_GET_SPECIFICATION opcode to retrieve an existing notification specification (**/config/notification_spec** object).

If you pass in an object POID, notification specification name, status, event name, or delivery description in the input flist, the opcode returns the appropriate **/config/notification_spec**. Otherwise, the opcode returns all **/config/notification_spec** objects in the BRM database.

The PCM_OP_NOTIFICATION_GET_SPECIFICATION opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION policy opcode to perform any customizations before retrieving the notification specification. See "Customizing Notification Retrieval".
- Calls the PCM_OP_SEARCH opcode to find the **/config/notification_spec** objects and prepares the output flist.
- Returns the POID of one or more **/config/notification_spec** objects in the PIN_FLD_RESULTS output flist array.

Managing In-Advance and Post-Expiration Notifications

Use the PCM_OP_NOTIFICATION_PROCESS_NOTIFICATION opcode to manage in-advance and post-expiration notifications. This opcode is called by the **pin_gen_notifications** utility.

PCM_OP_NOTIFICATION_PROCESS_NOTIFICATION is a wrapper opcode that calls other opcodes to create in-advance and post-expiration notifications. It does the following:

- Calls the PCM_OP_NOTIFICATION_VERIFY_PUBLISH_REQUIRED opcode to verify that the business event should be published. See "Verifying Customer Opt-In Preferences".
- Determines whether to send the notification now or schedule it for later. If the notification is scheduled for later, it creates a **/schedule/notification** object.
- Calls the opcode specified in the PIN_FLD_OPCODE input flist field, which generates the notification event.

> **Note:**
>
> The field passes in the opcode number. To find an opcode's number, see the opcode header files in *BRM_home*/**include/ops**.

Calculating the Notification Delivery Time

Use the PCM_OP_NOTIFICATION_CALC_DELIVERY_TIME opcode to calculate the customer's message delivery time. This opcode is called by the **pin_gen_notifications** utility.

The PCM_OP_NOTIFICATION_CALC_DELIVERY_TIME opcode calculates the delivery time based on the preferred time, scheduled time, silent period, and silent day settings. See "About Message Delivery Times" in *BRM Managing Customers* for more information about these settings. If the delivery time is scheduled for the future, the opcode returns the PIN_FLD_WHEN_T output flist field set to the delivery timestamp.

Verifying Customer Opt-In Preferences

Use the PCM_OP_NOTIFICATION_VERIFY_PUBLISH_REQUIRED opcode to verify a customer's opt-in or opt-out preferences for receiving notification messages.

The PCM_OP_NOTIFICATION_VERIFY_PUBLISH_REQUIRED opcode does the following:

- Checks the customer's opt-in and opt-out preferences by reading the PIN_FLD_NAME field set to the following in the customer's **/profile/subscriber_preferences** object: **NotifyOptInList** and **NotifyOptOutList**.

  If the **/profile/subscriber_preferences** object does not include the fields, it checks the value of the PIN_FLD_NOTIFY_OPT_TYPE field in the **/config/notification_spec** object.
- Returns the PIN_FLD_DELIVERY_STATUS output flist field set to one of the following:
  - **1**: Specifies *to publish* the business event to the Kafka server
  - **0**: Specifies *not to publish* the business event

Retrieving Last Notification Time and Offset

Use the PCM_OP_NOTIFICATION_GET_LASTNOTIFY_TSTAMP opcode to determine if an event's due date and time occur during the customer's message delivery window. This is a helper opcode that is called by internal opcodes before they create the actual notification event.

The PCM_OP_NOTIFICATION_GET_LASTNOTIFY_TSTAMP opcode returns the following fields in the PIN_FLD_DATES output flist array:

- PIN_FLD_DUE_T set to the event's due date and time
- PIN_FLD_STATUS set to one of the following:
  - **0**: Specifies that the due date timestamp *does not occur* during the delivery window
  - **1**: Specifies that the due date timestamp *does occur* during the delivery window

Customizing Notifications

You can customize notifications before they are sent to external applications by using the following opcodes:

- PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION. See "Customizing Notification Retrieval".
- PCM_OP_NOTIFICATION_POL_PREP_SPECIFICATION. See "Preparing Notification Specifications".
- PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION. See "Validating Notification Specifications".

Customizing Notification Retrieval

Use the PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION policy opcode to validate and implement any customizations while retrieving notification specifications. This policy opcode is called by the PCM_OP_NOTIFICATION_GET_SPECIFICATION opcode.

By default, the PCM_OP_NOTIFICATION_POL_GET_SPECIFICATION opcode does the following, but you can customize it to do more:

- Validates the mandatory parameters passed in the input flist.
- Returns errors if any parameters are invalid.

Preparing Notification Specifications

Use the PCM_OP_NOTIFICATION_POL_PREP_SPECIFICATION policy opcode to prepare the fields for the notification specification and implement any customizations while creating the notification specifications. This opcode is called by the PCM_OP_NOTIFICATION_CREATE_SPECIFICATION opcode.

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

Validating Notification Specifications

Use the PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION policy opcode to validate and implement any customizations while creating notification specifications. This policy opcode is called by the PCM_OP_NOTIFICATION_CREATE_SPECIFICATION opcode.

The PCM_OP_NOTIFICATION_POL_VALID_SPECIFICATION policy opcode does the following:

- Validates the mandatory parameters in the input flist.
- Calls the PCM_OP_SEARCH opcode to search for duplicate notification specification names.
- Validates that all fields are in the required format.
- Does one of the following:
  - If it encounters BRM system-related errors, sets the PIN_FLD_RESULT output flist field to FAIL, shares the designated error message, and clears the Error Buf to avoid any break in the flow.

- If it does not encounter errors, reviews other logical areas for failures and sets the PIN_FLD_RESULT output flist field to PASS.

19

Number Manager and SIM Card Manager Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) Number Manager and SIM Card Manager opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Number Manager Opcodes
- SIM Card Manager Opcodes

Opcodes Described in This Chapter

Table 19-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 19-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_SCHEDULE_CREATE	Managing Number Quarantine
PCM_OP_ACT_SCHEDULE_DELETE	Managing Number Quarantine
PCM_OP_CUST_POL_CANONICALIZE	Creating Blocks of Numbers
PCM_OP_DEVICE_ASSOCIATE	Customizing How Numbers Are Associated with Services
PCM_OP_DEVICE_CREATE	Creating Blocks of Numbers Customizing Telephone Number Attributes Creating SIM Cards Customizing SIM Card Validation
PCM_OP_DEVICE_POL_ASSOCIATE	Customizing SIM Card Service Association
PCM_OP_DEVICE_POL_DELETE	Deleting a Number
PCM_OP_DEVICE_SET_ATTR	Modifying Blocks of Numbers Managing Number Quarantine Customizing How a Number Can Be Changed

Table 19-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEVICE_SET_STATE	Managing Number Quarantine Customizing How Numbers Are Associated with Services Provisioning SIM Cards Customizing SIM Card Service Association
PCM_OP_NUM_CREATE_BLOCK	Creating Blocks of Numbers Customizing Number Normalization
PCM_OP_NUM_MODIFY_BLOCK	Creating Blocks of Numbers Modifying Blocks of Numbers
PCM_OP_NUM_POL_CANONICALIZE	Creating Blocks of Numbers Modifying Blocks of Numbers Customizing Number Normalization Customizing How a Number Can Be Changed
PCM_OP_NUM_POL_DEVICE_ASSOCIATE	Customizing How Numbers Are Associated with Services
PCM_OP_NUM_POL_DEVICE_CREATE	Customizing Telephone Number Attributes
PCM_OP_NUM_POL_DEVICE_DELETE	Deleting a Number
PCM_OP_NUM_POL_DEVICE_SET_ATTR	Customizing How a Number Can Be Changed
PCM_OP_NUM_PORT_IN	Managing Number Portability
PCM_OP_NUM_PORT_OUT	Managing Number Portability
PCM_OP_NUM_QUARANTINE	Managing Number Quarantine
PCM_OP_NUM_SPLIT_BLOCK	Splitting Blocks of Numbers
PCM_OP_SIM_CREATE_ORDER	Creating and Updating SIM Card Orders
PCM_OP_SIM_DEVICE_PROVISION	Provisioning SIM Cards
PCM_OP_SIM_POL_DEVICE_ASSOCIATE	Customizing SIM Card Service Association
PCM_OP_SIM_POL_DEVICE_CREATE	Customizing SIM Card Validation
PCM_OP_SIM_POL_DEVICE_SET_ATTR	Customizing SIM Card Validation Customizing SIM Card Number Changes
PCM_OP_SIM_POL_ORDER_CREATE	Creating SIM Cards
PCM_OP_SIM_PROCESS_ORDER_RESPONSE	Creating SIM Cards
PCM_OP_SIM_UPDATE_ORDER	Creating and Updating SIM Card Orders Creating SIM Cards

Number Manager Opcodes

See the following topics:

- Creating Blocks of Numbers
- Modifying Blocks of Numbers
- Splitting Blocks of Numbers
- Managing Number Quarantine
- Managing Number Portability

- Customizing Number Manager

Creating Blocks of Numbers

To create a block of numbers, use PCM_OP_NUM_CREATE_BLOCK. This opcode creates a **/block** object and the specified number of telephone numbers, created as **/device/num** objects.

The opcode identifies whether you want to modify the block or extend/shrink the block. If the value of the PIN_FLD_REQ_MODE flag is **True**, PCM_OP_NUM_MODIFY_BLOCK extends or shrinks the number block. If the value of the flag is **False**, PCM_OP_NUM_MODIFY_BLOCK modifies the number block.

PCM_OP_NUM_CREATE_BLOCK calls PCM_OP_CUST_POL_CANONICALIZE and PCM_OP_NUM_POL_CANONICALIZE to format the block name and telephone numbers. It calls PCM_OP_DEVICE_CREATE to create numbers and PCM_OP_CREATE_OBJ to create the block.

This opcode returns an error if the block name already exists or if any of the telephone numbers already exists.

Modifying Blocks of Numbers

To modify a block of numbers, use PCM_OP_NUM_MODIFY_BLOCK.

If numbers are changed, this opcode calls PCM_OP_NUM_POL_CANONICALIZE to normalize the number and PCM_OP_DEVICE_SET_ATTR to change attributes such as the manufacturer, model number, and description.

This opcode returns an error if the new block name already exists or if the range of numbers specified is not valid.

If this opcode runs successfully, it returns the POID of the modified block and an array of POIDs for new blocks.

Splitting Blocks of Numbers

To split a block of numbers, use PCM_OP_NUM_SPLIT_BLOCK.

The input includes the POID and name of the existing block and the start and end numbers for each block.

If this opcode runs successfully, it returns the POIDs of the original block and the new blocks.

If an invalid block or number attribute is found, this opcode returns an error to the error buffer.

Managing Number Quarantine

To quarantine numbers, use PCM_OP_NUM_QUARANTINE.

This opcode creates or deletes a **/schedule/device** object to manage the telephone number quarantine. PCM_OP_NUM_QUARANTINE is called by PCM_OP_DEVICE_SET_STATE when the number device state changes in and out of the Quarantined state. This is the default behavior as specified in the number device state configuration.

- If the state transition is to Quarantined, this opcode calls PCM_OP_ACT_SCHEDULE_CREATE to create a **/schedule/device** object that, at the end of the quarantine period, is used to change the state to Unassigned.

- There are two possible exit transitions from the Quarantined state:
  - If the state transition is from Quarantined to Assigned, PCM_OP_NUM_QUARANTINE calls PCM_OP_SEARCH to find the **/schedule/device** object associated with the number and calls PCM_OP_ACT_SCHEDULE_DELETE to delete the **/schedule/device** object.
  - If the state transition is from Quarantined to Unassigned, PCM_OP_NUM_QUARANTINE calls PCM_OP_DEVICE_SET_ATTR to delete the text in the device description field.

To customize how number quarantine works, you can edit and load the **pin_device_state** file to call a custom opcode instead of calling PCM_OP_NUM_QUARANTINE. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Managing Number Portability

To manage number portability, use PCM_OP_NUM_PORT_IN and PCM_OP_NUM_PORT_OUT.

PCM_OP_NUM_PORT_IN creates a number device using the provided number. This opcode is used when porting the number from another service provider into the existing network.

PCM_OP_NUM_PORT_IN calls opcodes to do the following:

- Create the number and the **/device/num** object.
- Set the status of the **/device/num** object.
- Reformats the number provided in the input flist to your required format.
- Validate the required information in the input flist.
- Search for the provided number in the database and, if found, return an error.
- If not found, create the number in the number device and update the status of the number device and commit the number to associate the device with the account.

PCM_OP_NUM_PORT_OUT sets the status of the specified telephone number as **quarantine_port_out** in the **/device/num** object. PCM_OP_NUM_PORT_OUT is used when porting the number from your network to another service provider.

> **Note:**
>
> The present state of the device must be assigned and the new state must be set to **quarantine_port_out**. If these are not the states, PCM_OP_NUM_PORT_IN returns an error.

PCM_OP_NUM_PORT_OUT calls opcodes to perform the necessary validations and set the status of the number device. This opcode:

- Validates the number to be ported out.
- Validates the current status of the number device.
- Updates the status of the number device.

Customizing Number Manager

See the following topics:

- Customizing Number Normalization
- Customizing How Numbers Are Associated with Services
- Customizing Telephone Number Attributes
- Customizing How a Number Can Be Changed

Customizing Number Normalization

To customize number normalization, use PCM_OP_NUM_POL_CANONICALIZE. This opcode handles number normalization when receiving numbers from applications and outputting numbers to other opcodes or applications.

This opcode is called by:

- PCM_OP_NUM_CREATE_BLOCK
- PCM_OP_NUM_MODIFY_BLOCK
- PCM_OP_NUM_PORT_IN

PCM_OP_NUM_POL_CANONICALIZE is called by PCM_OP_NUM_CREATE_BLOCK when a block of numbers is created.

You can customize this opcode to change normalization rules for handling numbers.

By default, this opcode performs the following translations:

- Adds two leading zero characters (00) if they are missing from the input.
- Removes all characters except digits.

PCM_OP_NUM_POL_CANONICALIZE returns an error when the minimum or maximum length is not valid. These values are defined in the PIN_NUM_CANON_MIN_LENGTH and PIN_NUM_CANON_MAX_LENGTH entries in the **pin_num.h** file in *BRM_home***/include**.

If successful, this opcode returns the normalized string.

Customizing How Numbers Are Associated with Services

To customize how numbers are associated with services, use PCM_OP_NUM_POL_DEVICE_ASSOCIATE. This opcode specifies the rules for associating or disassociate a number and a service. This opcode is called by PCM_OP_DEVICE_ASSOCIATE when a number is associated or disassociated with a service.

- If the number is being associated and there is no service currently associated with the number, this opcode calls PCM_OP_DEVICE_SET_STATE to change the state to Assigned. This is the default behavior as specified in the number device state configuration.
- If the number is being disassociated, and there is only one service associated with the number, this opcode calls PCM_OP_DEVICE_SET_STATE to change the state to Quarantined. This is the default behavior as specified in the number device state configuration.

When associating a number that is already associated with a service, the following rules are applied:

- A number can be associated with multiple services, but each service must be of a different type. For example, you cannot associate one number with two SMS services. To do so, you must customize PCM_OP_NUM_POL_DEVICE_ASSOCIATE.
- A number can be associated with multiple services, but all services must belong to the same account.
- If an associated service has a SIM card device associated with it, the number and the SIM card must have the same network element.

To find an account based on the IMSI or MSISDN in the CDR, PCM_OP_NUM_POL_DEVICE_ASSOCIATE copies the IMSI and MSISDN to the alias list array in the associated service object. The element ID is significant:

- The IMSI is copied to ALIAS_LIST[0]
- The MSISDN is copied to ALIAS_LIST[1]

PCM_OP_NUM_POL_DEVICE_ASSOCIATE returns an error in the following cases:

- The number is already associated with the same type of service. For example, if the number is already associated with a fax service, an error is returned if you try to associate the number with another fax service.
- The service that is being associated does not belong to the same account as an existing associated service.
- The number network element is not the same as the network element of a SIM card that is already associated with the service.

Customizing Telephone Number Attributes

To customize telephone number attributes when a number is created, use PCM_OP_NUM_POL_DEVICE_CREATE. This opcode validates a new number to make sure it is unique in the database.

This opcode is called by PCM_OP_DEVICE_CREATE and calls PCM_OP_NUM_POL_CANONICALIZE.

You can customize this opcode if you extend the number device attributes and require additional validation or if you want to change existing validations.

PCM_OP_NUM_POL_DEVICE_CREATE returns an error if a new number is not unique in the database.

Customizing How a Number Can Be Changed

To customize how a number can be changed, for example, which digits can be changed, use PCM_OP_NUM_POL_DEVICE_SET_ATTR.

By default, PCM_OP_NUM_POL_DEVICE_SET_ATTR supports changing US area codes by using the following logic: If the number starts with 001 and is 13 digits long, allow changing digits 4 through 6.

This opcode is called by PCM_OP_DEVICE_SET_ATTR and calls PCM_OP_NUM_POL_CANONICALIZE.

You can customize PCM_OP_NUM_POL_DEVICE_SET_ATTR to allow customized modifications on numbers. For example, you can customize this opcode to allow changing digits 5 through 7 of a number if the number starts with 44.

Deleting a Number

To delete a number, use PCM_OP_NUM_POL_DEVICE_DELETE. This opcode checks the state of the device. If the device state is PIN_NUM_STATE_NEW or PIN_NUM_STATE_UNASSIGNED, allows you to delete the device; otherwise, it generates an error and does not allow you to delete the device. This opcode is called by PCM_OP_DEVICE_POL_DELETE.

SIM Card Manager Opcodes

See the following topics:

- Creating and Updating SIM Card Orders
- Creating SIM Cards
- Provisioning SIM Cards
- Customizing SIM Card Manager

Creating and Updating SIM Card Orders

When you use SIM Card Administrator to create an order, PCM_OP_SIM_CREATE_ORDER creates an order object (**/order/sim)** in the database. It returns an error if it finds a duplicate SIM card number or IMSI. The order status is set to New.

When you update an order, PCM_OP_SIM_UPDATE_ORDER updates the order object in the database. It returns an error if it finds a duplicate SIM card number or IMSI.

PCM_OP_SIM_UPDATE_ORDER also changes the order status, for example, when you process a response file, or cancel an order.

PCM_OP_SIM_UPDATE_ORDER is called when a customer updates the order, or when the order status needs to be changed, for example, after processing a vendor response file. This opcode is also called when an order is canceled.

PCM_OP_SIM_UPDATE_ORDER checks for duplicate SIM card and IMSI numbers. If no error is found, this opcode updates a SIM card order object. If the order is being canceled, this opcode changes the status to Canceled. This opcode will not function if the order is not in the New status.

Creating SIM Cards

SIM Card Manager uses PCM_OP_SIM_PROCESS_ORDER_RESPONSE to process the order response file and create SIM card device objects (**/device/sim** objects) in the database.

PCM_OP_SIM_PROCESS_ORDER_RESPONSE does the following:

1. Reads the status of the order from the order object. If an order has been canceled, or there is no matching order for the response file, this opcode terminates and reports an error.
2. Finds the initial state of the device as defined in the **/config/device_state/sim** object. If the state is New, the SIM card is pre-provisioned. To disable pre-provisioning, or customize how SIM cards are provisioned, change the initial state from New to a customized state.
3. Finds the network element associated with the order.
4. Calls PCM_OP_DEVICE_CREATE to create the SIM card devices (**/device/sim**) in the BRM database.

5. Calls PCM_OP_SIM_UPDATE_ORDER to change the status of the order to either Partially Received or Received.

The vendor response file includes a list of SIM cards that you load into the BRM database by using SIM Administration Center.

To customize how SIM cards are validated during creation, use PCM_OP_SIM_POL_ORDER_CREATE.

Provisioning SIM Cards

SIM Card Manager uses PCM_OP_SIM_DEVICE_PROVISION to associate a SIM card with a service, and to disassociate the pre-provisioning service.

This opcode is called by PCM_OP_DEVICE_SET_STATE during the state transition from initial to new and from provisioning to release.

PCM_OP_SIM_DEVICE_PROVISION is used as the validation opcode for changing the device state from New to Provisioning and from Provisioning to Released or Failed Provisioning.

To customize pre-provisioning, you can write your own opcode and use your custom opcode as the validation opcode.

Customizing SIM Card Manager

See the following topics:

- Customizing SIM Card Service Association
- Customizing SIM Card Number Changes
- Customizing SIM Card Validation

Customizing SIM Card Service Association

Use PCM_OP_SIM_POL_DEVICE_ASSOCIATE to change how SIM cards and services are associated.

This opcode is called by PCM_OP_DEVICE_POL_ASSOCIATE.

- If the service is being associated, PCM_OP_SIM_POL_DEVICE_ASSOCIATE validates that the network element is the same as the network element for the number associated with the same service. This opcode then calls PCM_OP_DEVICE_SET_STATE to change the status from Released to Assigned.
- If the service is being unassociated, PCM_OP_SIM_POL_DEVICE_ASSOCIATE disassociates the service and calls PCM_OP_DEVICE_SET_STATE to change the status from Assigned to Unassigned.

To find an account based on the IMSI or MSISDN in the CDR, PCM_OP_SIM_POL_DEVICE_ASSOCIATE copies the IMSI and MSISDN to the alias list array in the associated service object. The element ID is significant:

- The IMSI is copied to ALIAS_LIST[0]
- The MSISDN is copied to ALIAS_LIST[1]

PCM_OP_SIM_POL_DEVICE_ASSOCIATE returns an error in the following cases:

- It returns an error if the network element of the SIM card and telephone number are not the same.
- It returns an error if the service type is already associated with the SIM card. The only exception is when two **/service/telco/gsm/telephony** services are associated with the same SIM card.
- It returns an error when the device and the service to be associated with it are not owned by the same account.

Customizing SIM Card Validation

Use PCM_OP_SIM_POL_DEVICE_CREATE to change validation rules for creating SIM card devices.

This opcode validates a device by validating the SIM card number, IMSI, KI, and network element values. For example, it makes sure that the numbers have the correct number of digits and are in the proper syntax. It also verifies that the SIM card does not already exist in the database.

This opcode is called by PCM_OP_DEVICE_CREATE when creating a SIM card device.

> **Note:**
>
> For information about validating the network element when updating a device, see PCM_OP_SIM_POL_DEVICE_SET_ATTR.

Customizing SIM Card Number Changes

Use PCM_OP_SIM_POL_DEVICE_SET_ATTR to change how SIM cards and services are associated.

This opcode ensures that the SIM card number (PIN_FLD_DEVICE_ID) cannot be changed.

This opcode is called by PCM_OP_DEVICE_POL_SET_ATTR when updating a SIM card device.

This opcode returns an error if an attempt is made to change SIM card number (PIN_FLD_DEVICE_ID).

20

Role Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) role opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Creating a Role
- Modifying a Role
- Deleting a Role
- Associating a pcm_client or admin_client Service with a Role

Opcodes Described in This Chapter

Table 20-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 20-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_CREATE_ROLE	Creating a Role
PCM_OP_CUST_UPDATE_ROLE	Modifying a Role
PCM_OP_CUST_DELETE_ROLE	Deleting a Role
PCM_OP_CUST_ASSOCIATE_ROLE	Associating a pcm_client or admin_client Service with a Role

For information about default users created during installation and their associated roles, see "Default users" in *System Administrator's Guide*. For information about the default roles that are created, see "Custom Users" in *System Administrator's Guide*.

Creating a Role

Use the PCM_OP_CUST_CREATE_ROLE opcode to create roles, defining the opcodes that can be run and the storable classes that can be read or updated by users with that role. This opcode must be run by a user with the **SuperUserRole** role. This opcode creates a **/config/role** object.

You create a role by including the following in the input flist:

- The name (PIN_FLD_NAME) and description (PIN_FLD_DESCR) of the new role and other identifying parameters.
- A PIN_FLD_PERMITTEDS element containing a list of either FM (PIN_FLD_FM_OPCODES) or base (PIN_FLD_BASE_OPCODES) opcodes that can be performed by the role. Do not include both.
- If you have included PCM_OP_EXEC_SPROC in PIN_FLD_BASE_OPCODES, you should provide a list of stored procedures that can be run by the role in PIN_FLD_CLASSES.
- For any other entry in PIN_FLD_BASE_OPCODES, you should provide a list of classes that can be modified by the role in PIN_FLD_CLASSES.

The PCM_OP_CUST_CREATE_ROLE opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Validates that either PIN_FLD_FM_OPCODES or PIN_FLD_BASE_OPCODES are present but not both (or if both are provided, that one is empty).
3. Validates that if the PIN_FLD_BASE_OPCODES field is populated, PIN_FLD_CLASSES is also provided and not empty.
4. Validates that none of the entries duplicate each other.
5. Creates a **/config/role** object with the supplied information.

Following is a sample input flist for the PCM_OP_CUST_CREATE_ROLE opcode. The sample below allows users with the **testnap_role** role to run the following:

- FM opcodes: Users can run the PCM_OP_CUST_CREATE_CUSTOMER and PCM_OP_BILL_MAKE_BILL opcodes, and any opcodes that start with "PCM_OP_BAL."
- Policy opcodes: Users can run PCM_OP_BAL_POL_VAL_BAL_TRANSFER as an entry level opcode because PIN_FLD_FM_OPCODES allows opcodes that start with "PCM_OP_BAL." It could also be allowed by including PCM_OP_BAL_POL_VAL_BAL_TRANSFER in PIN_FLD_FM_OPCODES directly.
- Base opcodes: Users can run the PCM_OP_WRITE_FLDS, PCM_OP_READ_FLDS, and PCM_OP_CREATE_OBJ opcodes on **/account** and **/bill** class objects and **/service** and its child class objects.

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role -1 0
0 PIN_FLD_ACCOUNT_OBJ           POID [0] 0.0.0.1 /account 123456 0
0 PIN_FLD_DESCR                 STR [0] "Permissions for testnap"
0 PIN_FLD_HOSTNAME              STR [0] "-"
0 PIN_FLD_NAME                  STR [0] "testnap role"
0 PIN_FLD_PROGRAM_NAME          STR [0] "testnap"
0 PIN_FLD_PERMITTEDS    ARRAY [0] allocated 20, used 2
1     PIN_FLD_FM_OPCODES        STR [0] "PCM_OP_CUST_CREATE_CUSTOMER,
PCM_OP_BILL_MAKE_BILL, PCM_OP_BAL_*"
1      PIN_FLD_BASE_OPCODES       STR [0] ""
1     PIN_FLD_CLASSES             STR [0] ""
0 PIN_FLD_PERMITTEDS    ARRAY [1] allocated 20, used 2
1     PIN_FLD_FM_OPCODES        STR [0] ""
1      PIN_FLD_BASE_OPCODE        STR [0] "PCM_OP_WRITE_FLDS"
1     PIN_FLD_CLASSES             STR [0] "/account,/bill,/service/*"
0 PIN_FLD_PERMITTEDS    ARRAY [2] allocated 20, used 2
1     PIN_FLD_FM_OPCODES        STR [0] ""
```

```
1      PIN_FLD_BASE_OPCODE        STR [0] "PCM_OP_READ_FLDS,PCM_OP_CREATE_OBJ"
1     PIN_FLD_CLASSES             STR [0] "/account,/bill,/service/*"
```

Following is the output flist for the PCM_OP_CUST_CREATE_ROLE opcode, containing the POID of the new **/config/role** object:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role 234567 0
```

Modifying a Role

Use the PCM_OP_CUST_UPDATE_ROLE opcode to modify roles, adding opcodes that can be run and storable classes that can be read or updated by users with that role. This opcode must be run by a user with the **SuperUserRole** role. This opcode modifies a **/config/role** object.

You add opcodes and storable classes to a role by including the following in the input flist:

- The POID of the **/config/role** object.
- A PIN_FLD_PERMITTEDS element containing a list of either FM (PIN_FLD_FM_OPCODES) or base (PIN_FLD_BASE_OPCODES) opcodes that can be performed by the role. Do not include both.
- If you have included PCM_OP_EXEC_SPROC in PIN_FLD_BASE_OPCODES, you should provide a list of stored procedures that can be run by the role in PIN_FLD_CLASSES.
- For any other entry in PIN_FLD_BASE_OPCODES, you should provide a list of classes that can be modified by the role in PIN_FLD_CLASSES.

The PCM_OP_CUST_UPDATE_ROLE opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Validates that if the PIN_FLD_BASE_OPCODES field is populated, PIN_FLD_CLASSES is also provided and not empty.
3. Validates that none of the entries duplicate each other or entries already contained in the role.
4. Updates the **/config/role** object with the supplied information. If an element (for example, PIN_FLD_FM_OPCODES) already exists on the object, the new entries will be appended to it. If the element does not exist, it will be added.

Following is a sample input flist for the PCM_OP_CUST_UPDATE_ROLE opcode. The sample below adds permissions for users with the **testnap_role** role to run the following:

- Base opcodes: Users can run the PCM_OP_WRITE_FLDS, PCM_OP_CREATE_OBJ, and PCM_OP_READ_FLDS opcodes on **/balance_group** and PCM_OP_READ_FLDS on **/group/sharing** in addition to whatever was defined for the role before.

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role 234567 0
0 PIN_FLD_PERMITTEDS    ARRAY [1] allocated 20, used 2
1     PIN_FLD_FM_OPCODES        STR [0] ""
1      PIN_FLD_BASE_OPCODE        STR [0]
"PCM_OP_WRITE_FLDS,PCM_OP_CREATE_OBJ"
1     PIN_FLD_CLASSES             STR [0] "/balance_group"
0 PIN_FLD_PERMITTEDS    ARRAY [2] allocated 20, used 2
1     PIN_FLD_FM_OPCODES        STR [0] ""
```

```
1      PIN_FLD_BASE_OPCODE         STR [0] "PCM_OP_READ_FLDS"
1     PIN_FLD_CLASSES              STR [0] "/balance_group,/group/sharing"
```

Following is the output flist for the PCM_OP_CUST_UPDATE_ROLE opcode, containing the POID of the modified **/config/role** object:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role 234567 0
```

Deleting a Role

Use the PCM_OP_CUST_DELETE_ROLE opcode to delete a role. This opcode deletes a **/config/role** object.

You delete a role by including the following in the input flist:

- The POID of the **/config/role** object.

The PCM_OP_CUST_DELETE_ROLE opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Deletes the **/config/role** object.

Following is a sample input flist for the PCM_OP_CUST_DELETE_ROLE opcode. The sample below deletes the role with the POID 234567:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role 234567 0
```

Following is the output flist for the PCM_OP_CUST_DELETE_ROLE opcode, containing the POID of the deleted **/config/role** object:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /config/role 234567 0
```

Associating a pcm_client or admin_client Service with a Role

Use the PCM_OP_CUST_ASSOCIATE_ROLE opcode to associate a pcm_client or admin_client service with an existing role. This opcode updates a **/service/pcm_client** or **/service/admin_client** object.

You associate a pcm_client or admin_client service with a role by including the following in the input flist:

- The POID of the pcm_client or admin_client service.
- The role that is being associated with the service (PIN_FLD_CONFIG_PROFILE_OBJ).

The PCM_OP_CUST_ASSOCIATE_ROLE opcode does the following:

1. Validates the mandatory parameters in the input flist.
2. Validates that the service is a .pcm_client or admin_client service.
3. Validates that the provided role exists.
4. Adds the role to the service in the PIN_FLD_CONFIG_ROLE element.

Following is a sample input flist for the PCM_OP_CUST_ASSOCIATE_ROLE opcode. The sample below associates the role with the POID 234567 with a pcm_client service:

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /service/pcm_client
123456 5
0 PIN_FLD_CONFIG_ROLE                    SUBSTRUCT [0] allocated 3, used 1
1     PIN_FLD_CONFIG_PROFILE_OBJ    POID [0] 0.0.0.1 /config/role 234567 0
```

Following is the output flist for the PCM_OP_CUST_ASSOCIATE_ROLE opcode, containing the POID of the modified **/service/pcm_client** object:

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /service/pcm_client
123456 5
```

21

Payment Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) payment opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Collecting Payments
- BRM-Initiated Payment Processing
- Externally Initiated Payment Processing
- Receiving Payments
- Validating Payments
- Configuring Unconfirmed Payment Processing
- Handling Overpayments and Underpayments
- Calculating Payment Collection Dates
- How BRM Selects the Items to Which Payments Are Applied
- Allocating Account Payments to Multiple Bill Units
- Reversing Payments
- Refunding Payments
- Writing Off Payments
- Finding Payment Info
- Adding a Custom Payment Method
- How Opcodes Read the Payment Method
- Processing Payment Fees
- Processing Payment Incentives
- Managing Top-Ups
- Suspending and Recycling Payments
- How BRM Handles Mandate Information for SEPA Processing

Opcodes Described in This Chapter

Table 21-1 lists the opcodes described in this chapter.

Caution:

- Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
- Do not use SQL commands to change data in the database. Always use the API.

Table 21-1 Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_POL_EVENT_NOTIFY	Triggering Top-ups
PCM_OP_ACT_USAGE	BRM-Initiated Payment Processing Receiving Payments
PCM_OP_AR_ACCOUNT_ADJUSTMENT	How BRM Handles Manual Standard Top-Ups How BRM Handles Automatic Standard Top-Ups
PCM_OP_AR_ACCOUNT_WRITEOFF	Writing Off Payments
PCM_OP_AR_BILL_WRITEOFF	Writing Off Payments
PCM_OP_AR_ITEM_WRITEOFF	Writing Off Payments
PCM_OP_AR_REVERSE_WRITEOFF	Applying Payment Fees
PCM_OP_BILL_ITEM_REFUND	Refunding Payments
PCM_OP_BILL_ITEM_TRANSFER	BRM-Initiated Payment Processing Triggering Payment Incentives
PCM_OP_BILL_MAKE_BILL	Granting Payment Incentives
PCM_OP_BILL_POL_CALC_PYMT_DUE_T	Calculating Payment Collection Dates
PCM_OP_BILL_RCV_PAYMENT	BRM-Initiated Payment Processing Externally Initiated Payment Processing Receiving Payments Allocating Account Payments to Multiple Bill Units How BRM Handles Manual Standard Top-Ups
PCM_OP_BILL_REVERSE	Reversing Payments How Payments Are Recycled to and from Suspense Retrieving Recycled Payments How Suspended Payments Are Reversed How Payments Are Reversed During Recycling How Payments Are Removed As Unallocatable
PCM_OP_BILL_REVERSE_PAYMENT	BRM-Initiated Payment Processing Reversing Payments Reversing Payment Incentives How Suspended Payments Are Reversed How Payments Are Reversed During Recycling How Payments Are Removed As Unallocatable
PCM_OP_BILL_SET_LIMIT_AND_CR	Setting a Customer's Automatic Top-Up Threshold

Table 21-1 (Cont.) Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_TRANSFER_BALANCE	How BRM Handles Manual Sponsored Top-Ups How BRM Handles Automatic Sponsored Top-Ups About Transferring Sponsored Top-Ups from Debit Balances
PCM_OP_CUST_AMEND_CREDITOR_INFO	How BRM Handles Mandate Information for SEPA Processing Updating Creditor Information
PCM_OP_CUST_AMEND_MANDATE	How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_CANCEL_MANDATE	How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_COMMIT_CUSTOMER	Processing Credit Card Information during Account Creation Implementing Automatic Standard Top-Ups Implementing Manual Sponsored Top-Ups Implementing Automatic Sponsored Top-Ups
PCM_OP_CUST_CREATE_PAYINFO	How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_CREATE_TOPUP	Setting Up Top-Up Information in an Account Preparing an Account's Top-Up Information Validating an Account's Top-Up Information Modifying an Account's Top-Up Information
PCM_OP_CUST_UPDATE_CUSTOMER	Implementing Automatic Standard Top-Ups Implementing Manual Sponsored Top-Ups Implementing Automatic Sponsored Top-Ups
PCM_OP_CUST_DELETE_PAYINFO	How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_DELETE_TOPUP	Deleting Top-Ups
PCM_OP_CUST_FIND_PAYINFO	Finding Payment Info
PCM_OP_CUST_MODIFY_TOPUP	Setting Up Top-Up Information in an Account Preparing an Account's Top-Up Information Validating an Account's Top-Up Information Modifying an Account's Top-Up Information
PCM_OP_PYMT_POL_OVER_PAYMENT	Handling Overpayments and Underpayments
PCM_OP_CUST_POL_PREP_NAMEINFO	About the Suspense Account
PCM_OP_CUST_POL_PREP_PAYINFO	Adding a Custom Payment Method
PCM_OP_CUST_POL_PREP_TOPUP	Preparing an Account's Top-Up Information Validating an Account's Top-Up Information Finding Sponsored Top-Up Groups
PCM_OP_PYMT_POL_UNDER_PAYMENT	Handling Overpayments and Underpayments
PCM_OP_CUST_POL_VALID_PAYINFO	Validating Credit Card and Direct Debit Transactions Validating an Account's Top-Up Information How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_PREP_CUSTOMER	Validating Credit Card and Direct Debit Transactions

Table 21-1 (Cont.) Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_SET_NAMEINFO	About the Suspense Account
PCM_OP_CUST_SET_PAYINFO	How BRM Handles Mandate Information for SEPA Processing
PCM_OP_CUST_SET_STATUS	BRM-Initiated Payment Processing
PCM_OP_CUST_SET_TOPUP	Setting Up Top-Up Information in an Account Finding Sponsored Top-Up Groups Implementing Automatic Standard Top-Ups Implementing Manual Sponsored Top-Ups Implementing Automatic Sponsored Top-Ups Setting an Account's Sponsored Top-Up Member Status and PIN
PCM_OP_PYMT_APPLY_FEE	BRM-Initiated Payment Processing Externally Initiated Payment Processing Applying Payment Fees
PCM_OP_PYMT_CHARGE	BRM-Initiated Payment Processing Processing Credit Card Charges Processing Direct Debit Charges Checking the Results of BRM-Initiated Batch Payment Operations
PCM_OP_PYMT_CHARGE_CC	BRM-Initiated Payment Processing Processing Credit Card Charges Processing a Batch of Direct Debit Charges
PCM_OP_PYMT_CHARGE_DD	BRM-Initiated Payment Processing Processing Direct Debit Charges About Paymentech Direct Debit Implementation Creating a Custom Direct Debit Implementation Processing a Batch of Direct Debit Charges
PCM_OP_PYMT_CHARGE_DDEBIT	About Paymentech Direct Debit Implementation

Table 21-1 (Cont.) Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_PYMT_COLLECT	Collecting Payments BRM-Initiated Payment Processing Externally Initiated Payment Processing Receiving Payments Validating Payments About the Default Payment Validation Process About Payment Validation Flags How BRM Selects the Items to Which Payments Are Applied Configuring Unconfirmed Payment Processing Allocating Account Payments to Multiple Bill Units Allocating Externally Initiated Payments by Due Amount Refunding Payments Checking the Results of BRM-Initiated Batch Payment Operations Applying Payment Fees Suspending Payments during Payment Processing How Payments Are Recycled to and from Suspense
PCM_OP_PYMT_FIND_TOPUP_EVENTS	Finding Top-Up Events Viewing Sponsored Top-Up History
PCM_OP_PYMT_GET_ACH_INFO	Processing Credit Card Information during Account Creation
PCM_OP_PYMT_GRANT_INCENTIVE	Processing Payment Incentives Triggering Payment Incentives Granting Payment Incentives
PCM_OP_PYMT_ITEM_TRANSFER	Externally Initiated Payment Processing Receiving Payments
PCM_OP_PYMT_MBI_DISTRIBUTE	Allocating Account Payments to Multiple Bill Units
PCM_OP_PYMT_MBI_ITEM_SEARCH	Allocating Account Payments to Multiple Bill Units
PCM_OP_PYMT_POL_APPLY_FEE	Applying Payment Fees Customizing Payment Fees
PCM_OP_PYMT_POL_CHARGE	BRM-Initiated Payment Processing Customizing Payment Failure Reason Codes
PCM_OP_PYMT_POL_COLLECT	BRM-Initiated Payment Processing Externally Initiated Payment Processing Processing the Results of Credit Card Transactions
PCM_OP_PYMT_POL_GRANT_INCENTIVE	Granting Payment Incentives
PCM_OP_PYMT_POL_MBI_DISTRIBUTE	Allocating Account Payments to Multiple Bill Units
PCM_OP_PYMT_POL_OVER_PAYMENT	Externally Initiated Payment Processing How BRM Selects the Items to Which Payments Are Applied
PCM_OP_PYMT_POL_PRE_COLLECT	BRM-Initiated Payment Processing Externally Initiated Payment Processing Allocating Account Payments to Multiple Bill Units
PCM_OP_PYMT_POL_PROVISION_INCENTIVE	Triggering Payment Incentives

Table 21-1 (Cont.) Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_PYMT_POL_PURCHASE_DEAL	How BRM Handles Manual Standard Top-Ups How BRM Handles Automatic Standard Top-Ups Offering Discount Incentives with Top-Ups
PCM_OP_PYMT_POL_SPEC_COLLECT	Processing Credit Card Information during Account Creation
PCM_OP_PYMT_POL_SPEC_VALIDATE	Validating Credit Card and Direct Debit Transactions
PCM_OP_PYMT_POL_SUSPEND_PAYMENT	Validating Payments Allocating Account Payments to Multiple Bill Units Customizing Suspended Payment Distribution
PCM_OP_PYMT_POL_UNDER_PAYMENT	Externally Initiated Payment Processing How BRM Selects the Items to Which Payments Are Applied
PCM_OP_PYMT_POL_VALIDATE	Validating Credit Card and Direct Debit Transactions
PCM_OP_PYMT_POL_VALIDATE_PAYMENT	Validating Payments About the Default Payment Validation Process About Payment Validation Flags Allocating Account Payments to Multiple Bill Units Allocating Externally Initiated Payments by Due Amount Configuring Unconfirmed Payment Processing Processing Paymentech Address Validation Return Codes Suspending Payments during Payment Processing Customizing Payment Suspense Validation
PCM_OP_PYMT_POL_VALID_VOUCHER	How BRM Handles Manual Standard Top-Ups
PCM_OP_PYMT_PROVISION_INCENTIVE	Processing Payment Incentives
PCM_OP_PYMT_RECOVER	Checking the Results of BRM-Initiated Batch Payment Operations
PCM_OP_PYMT_RECOVER_CC	Checking the Results of BRM-Initiated Batch Payment Operations
PCM_OP_PYMT_RECOVER_DD	Checking the Results of BRM-Initiated Batch Payment Operations
PCM_OP_PYMT_RECYCLE_PAYMENT	Allocating Account Payments to Multiple Bill Units How Payments Are Recycled to and from Suspense Retrieving Recycled Payments How Payments Are Reversed During Recycling
PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH	Reversing Payments How Payments Are Recycled to and from Suspense Retrieving Recycled Payments
PCM_OP_PYMT_REVERSE_INCENTIVE	Reversing Payments Reversing Payment Incentives

Table 21-1 (Cont.) Payment Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_PYMT_SELECT_ITEMS	BRM-Initiated Payment Processing Externally Initiated Payment Processing About Payment Validation Flags How BRM Selects the Items to Which Payments Are Applied Handling Overpayments and Underpayments Allocating Account Payments to Multiple Bill Units Allocating Externally Initiated Payments by Due Amount Applying Payment Fees
PCM_OP_PYMT_TOPUP	Triggering Top-ups How BRM Handles Manual Standard Top-Ups How BRM Handles Automatic Standard Top-Ups How BRM Handles Manual Sponsored Top-Ups How BRM Handles Automatic Sponsored Top-Ups Implementing Manual Standard Top-Ups Offering Discount Incentives with Top-Ups About Transferring Sponsored Top-Ups from Debit Balances
PCM_OP_PYMT_VALIDATE	Validating Credit Card and Direct Debit Transactions Processing Credit Card Information during Account Creation
PCM_OP_PYMT_VALIDATE_CC	Validating Credit Card and Direct Debit Transactions
PCM_OP_PYMT_VALIDATE_DD	Validating Credit Card and Direct Debit Transactions
PCM_OP_PYMT_VALIDATE_PAYMENT	Externally Initiated Payment Processing Validating Payments About the Default Payment Validation Process Allocating Account Payments to Multiple Bill Units Configuring Unconfirmed Payment Processing Suspending Payments during Payment Processing How Payments Are Recycled to and from Suspense

Collecting Payments

The main payment collection opcode is PCM_OP_PYMT_COLLECT. This opcode communicates with a payment processor to perform payment collections, refunds, and reversals. It processes payments according to the payment processing type: BRM-initiated, or externally initiated.

PCM_OP_PYMT_COLLECT allocates the payment to open items for each bill unit specified for the account. This opcode calls other opcodes to validate payments and calls various policy opcodes that allow you to customize payment collection.

By default, BRM allocates payments automatically. You can prevent allocation by using the PIN_BILLFLG_DEFER_ALLOCATION flag (0x1000000) in the PCM_OP_PYMT_COLLECT input flist PIN_FLD_CHARGES array.

Finding Payments

PCM_OP_PYMT_ITEM_SEARCH searches for **/item** objects with a variable number of input parameters. This opcode calls PCM_OP_SEARCH based on the input argument fields.

For collective bills, PCM_OP_PYMT_ITEM_SEARCH checks the **RejectPaymentsForPreviousBill** business parameter.

If **RejectPaymentsForPreviousBill** is enabled, the opcode does not accept the payment. BRM sends such a payment to the suspense payment account.

If the **RejectPaymentsForPreviousBill** is disabled, the opcode can accept the payment. In such a case, the opcode searches for the bill in the **history_bills** objects. If the bill is located in the **history_bills** objects, the payment is accepted for the bill with the same POID as the bill being processed. If such a bill is not located in **history_bills** objects, the payment is sent to a suspense payment account.

BRM-Initiated Payment Processing

For BRM-initiated payment processing, the normal flow of PCM_OP_PYMT_COLLECT is as follows:

1. Creates a batch checkpoint.
2. Calls PCM_OP_PYMT_SELECT_ITEMS to identify the items to which the payments are applied. See "How BRM Selects the Items to Which Payments Are Applied".
3. Calls PCM_OP_PYMT_POL_PRE_COLLECT, to perform policy checks before the charge, payment, or refund occurs after allocating the PIN_FLD_CHARGE array elements to open items.

   You can change the minimum credit card charge amount by modifying the default minimum payment amount in PCM_OP_PYMT_POL_PRE_COLLECT.

   You can also customize PCM_OP_PYMT_POL_PRE_COLLECT to retrieve soft descriptor information that enables you to display the name under which you do business (your DBA name), charge offer name, and customer service number on your customer's checking account or credit card statement.
4. Calls PCM_OP_PYMT_CHARGE to perform a BRM-initiated payment transaction.

   PCM_OP_PYMT_COLLECT checks the **/config/payment** object to determine which opcode to call to retrieve the payment information from the Payment DM and to create the charge: PCM_OP_PYMT_CHARGE_CC or PCM_OP_PYMT_CHARGE_DD.
5. For SEPA payments, creates the SEPA Direct Debit payment request (**/sepa/dd**).
6. Calls PCM_OP_BLL_RCV_PAYMENT. PCM_OP_BLL_RCV_PAYMENT:
   a. Calls PCM_OP_ACT_USAGE to update the **/event/billing/payment/** object.
   b. Calls PCM_OP_BILL_ITEM_TRANSFER to allocate each payment to open items for each bill unit (**/billinfo** object) specified for the account.
7. Calls PCM_OP_PYMT_APPLY_FEE to apply payment fees. If a payment fee is applied, the POID of the payment fee event is added to the PCM_OP_PYMT_APPLY_FEE output flist.
8. Calls PCM_OP_PYMT_POL_COLLECT, to process the result of a credit card transaction for a specified account bill unit. PCM_OP_PYMT_POL_COLLECT sends the charge result

to the policy opcode which specifies both the result to be returned to the caller and the actions to be performed on the account's bill unit.

For BRM-initiated payments, PCM_OP_PYMT_POL_COLLECT calls PCM_OP_CUST_SET_STATUS if the operation requires a status change.

9. By using checkpoints, finalizes the batch.

> **Note:**
>
> The BRM-initiated payment collection utilities (**pin_collect** and **pin_deposit**) process payments for multiple bills associated with multiple bill units. To process externally initiated payments, such as checks and cash payments, for multiple bill units, write custom code that records the deposits you have received and calls PCM_OP_PYMT_COLLECT.

When PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_CHARGE, the input flist contains a PIN_FLD_SESSION_OBJ field, which defines the session in which the event occurred: either a payment batch event or a refund batch event.

The input flist also contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified in each operation, so a single batch can contain a mixture of different commands.

The billing information for each operation, such as credit card number and expiration date, can be specified as options on the input flist. If it isn't specified, PCM_OP_PYMT_CHARGE retrieves the necessary information from the account objects specified in the operations. After it has all the data, the operations are forwarded to the opcode responsible for processing that payment method (for example, PCM_OP_PYMT_CHARGE_CC and PCM_OP_PYMT_CHARGE_DD for credit card charges and direct debit charges, respectively).

> **Note:**
>
> For security reasons, the credit card CVV2 and CID numbers are not stored in BRM. If the **CCcollect** business parameter is enabled, PCM_OP_PYMT_CHARGE passes the security information to the payment processor for authorization and collection only at time of account creation. When collecting payments, PCM_OP_PYMT_CHARGE does not pass the information. In addition, it omits the PIN_FLD_SECURITY_ID field from the PCM_OP_ACT_USAGE input flist so it is not written to the **/event/billing/charge/cc** object. The result is that the CVV2/CID information is stored in the database with a **NULL** value.

Each of the commands listed below in Table 21-2 can be run as an operation.

Table 21-2 Commands to Perform Transactions

Command	Description
PIN_CHARGE_CMD_VERIFY	Verify the address information.
PIN_CHARGE_CMD_AUTH_ONLY	Authorize a charge. The credit limit is decreased on the credit card, but no charge is posted.
PIN_CHARGE_CMD_CONDITION	Authorize and deposit a charge.

Table 21-2 (Cont.) Commands to Perform Transactions

Command	Description
PIN_CHARGE_CMD_DEPOSIT	Deposit a previously authorized charge.
PIN_CHARGE_CMD_RECOVER_PAYMENT	Recovers payments by using outstanding checkpoints.
PIN_CHARGE_CMD_REFUND	Refund a charge.
PIN_CHARGE_CMD_RESUBMIT	Resubmit the batch of charges.
PIN_CHARGE_CMD_RFR	For transactional payments, requests the RFR file to retrieve payments. For nontransactional payments, reads the RFR file to retrieve payments.

For each operation specified, the result of the operation is stored in the corresponding **/event/billing/charge** object.

The result is stored both as a numeric value returned by the payment processor, and as an enumerated result. The enumerated result should generally be used by applications to determine what happened because its values are independent of which settlement house was used.

Possible result values for an operation are shown in Table 21-3:

Table 21-3 Result Values for Operation

Input PIN_FLD_RESULT	Output PIN_FLD_RESULT	Output PIN_FLD_DESCR
PIN_CHARGE_RES_PASS	PIN_RESULT_PASS	Validation successful
PIN_CHARGE_RES_SRVC_UNAVAIL	PIN_RESULT_PASS	Validation successful
PIN_CHARGE_RES_FAIL_ADDR_AVS	PIN_RESULT_FAIL	Unable to verify address
PIN_CHARGE_RES_FAIL_ADDR_LOC	PIN_RESULT_PASS	Street address mismatch
PIN_CHARGE_RES_FAIL_ADDR_ZIP	PIN_RESULT_FAIL	ZIP code mismatch
PIN_CHARGE_RES_FAIL_CARD_BAD	PIN_RESULT_FAIL	Credit card number not valid
PIN_CHARGE_RES_FAIL_DECL_SOFT	PIN_RESULT_FAIL	General soft decline Card failed credit check
PIN_CHARGE_RES_FAIL_DECL_HARD	PIN_RESULT_FAIL	General hard decline Card failed credit check
PIN_CHARGE_RES_FAIL_NO_ANS	PIN_RESULT_FAIL	No answer from settlement house Unable to validate now
PIN_CHARGE_RES_CHECKPOINT	PIN_RESULT_FAIL	Unable to validate now
PIN_CHARGE_RES_CVV_BAD	PIN_RESULT_FAIL	Security ID check failed

General soft declines are failures that can be retried later with possible success. This includes reasons like insufficient credit limit and other transitory causes. General hard declines are failures that are unlikely to succeed if retried. These include reasons like lost and stolen credit card and chronic payment failures.

> **Note:**
>
> By default, account balances are not updated if there is a decline. To update account balances when a decline occurs, you must customize PCM_OP_PYMT_POL_CHARGE.

You can send multiple charges in one call by using the PIN_FLD_CHARGES array on the input flist. This array is designed for batch processing; you just make one call to PCM_OP_PYMT_CHARGE for a whole list of charges (or accounts to charge). The following entries on the PCM_OP_PYMT_CHARGE input flist are of special interest:

- The PIN_FLD_POID entry at the top of the input flist is only used for routing; it only requires a correct (user) database number.
- The PIN_FLD_ACCOUNT_OBJ entry is the POID of the account actually being charged (verified) by this element of the PIN_FLD_CHARGES array. In a batch, this POID is presumably different for every element.

If the PCM_OPFLG_CALC_ONLY flag is set, the opcode does not change any fields in the database and does not create an **/event/billing/charge** object. However, it does provide a charge calculation to the caller by returning the fields that would have been used to create the event object and the charge item.

> **Note:**
>
> Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

Initiating a Charge

PCM_OP_PYMT_CHARGE performs a BRM-initiated payment transaction.

PCM_OP_PYMT_CHARGE is called by PCM_OP_PYMT_COLLECT, and is the main entry point opcode for all BRM-initiated payment activities.

The input flist contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified in each operation, so a single batch can contain a mixture of different commands.

PCM_OP_PYMT_CHARGE calls the opcode responsible for processing the relevant payment method (for example, PCM_OP_PYMT_CHARGE_CC and PCM_OP_PYMT_CHARGE_DD for credit card charges and direct debit charges, respectively).

PCM_OP_PYMT_CHARGE works as follows:

1. Opens a transaction.
2. Using checkpoints, creates the **/event/billing/charge** event for each payment.
3. Calls PCM_OP_PYMT_CHARGE_CC or PCM_OP_PYMT_CHARGE_DD, depending on the payment method, to process the charges from the payment DM.

4. Updates the checkpoint in the charge event for each transaction received from the payment DM.
5. For each PIN_FLD_CHARGES in the input flist:
   a. Closes billing for the account.
   b. Creates a payment item and records the account number, bill number, and transaction ID from the input flist, and the fact that the money has been received.

      > **Important:**
      >
      > The maximum length of a payment transaction ID is 16 characters for BRM-initiated payments and 30 characters for externally initiated payments. If your company generates transaction IDs by appending characters to the payment batch IDs, ensure that the batch IDs are short enough to be within the limit after the additional characters are appended. If the payment transaction ID does not comply with the length restrictions, BRM does not generate an ID for the payment. (BRM-initiated payment transaction IDs are restricted by Paymentech processing requirements.)

   c. Adds the PIN_FLD_STATUS value in the output flist.

      For failed payments, sets the PIN_FLD_STATUS value to **PIN_PYMT_FAILED**. For successful payments, sets the PIN_FLD_STATUS value to **PIN_PYMT_SUCCESS**.

   d. Calls PCM_OP_PYMT_POL_CHARGE to update the reasons array. PCM_OP_PYMT_POL_CHARGE provides the ability to map the online and offline payment result to the payment status and the reason IDs defined in the **/strings** object. In the output flist PIN_FLD_REASONS array, the array of PIN_FLD_REASON_ID fields contains the failure reasons sent by the payment processor. You can configure PCM_OP_PYMT_POL_CHARGE to apply fees for failed credit card and direct debit transactions based on the reason for failure.
   e. Sends the payment status and the reasons array (PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID) to PCM_OP_PYMT_COLLECT.
   f. For payments with a successful status, applies the charge to the customer's account. For payments with a failed status, sends the PIN_FLD_STATUS value to PCM_OP_PYMT_APPLY_FEE to create the payment fees.
6. Closes the transaction.

Processing Credit Card Charges

To perform a credit card charge, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_CC.

This opcode supports all commands handled by PCM_OP_PYMT_CHARGE.

The PCM_OP_PYMT_CHARGE_CC input flist contains the PIN_FLD_SESSION_OBJ field, which references either the **/event/billing/batch/payment** or **/event/billing/batch/refund** object. This determines the batch type being submitted (payment or refund).

The PCM_OP_PYMT_CHARGE_CC input flist contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified in each operation, so a single batch can contain a mixture of different commands. This opcode supports all commands handled by PCM_OP_PYMT_CHARGE.

The operations are forwarded to the credit card processing Data Manager (for example, the Paymentech DM) for processing.

With most credit card payment services, performing an authorization is much faster than a conditional deposit (authorization plus deposit). Thus, for applications where latency is important, it may be desirable to perform just the authorization step in real-time. BRM daily billing performs the necessary deposits for all outstanding authorizations from the previous day. This removes a significant amount of latency from the real-time process, but still authorizes the charge so it is guaranteed to deposit successfully.

The set of Paymentech return codes handled by BRM is listed in the *BRM_home***/sys/dm_fusa/fusa_codes** file.

Unless the PCM_OPFLG_CALC_ONLY flag is specified, PCM_OP_PYMT_CHARGE_CC creates an **/event/billing/charge/cc** object for each operation. If an array of operations was specified, then more than one event object is created. The event objects are created even if the credit card operations cannot be performed.

> **Note:**
>
> Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example, Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

Processing the Results of Credit Card Transactions

To process the result of a credit card transaction for a specified account, use PCM_OP_PYMT_POL_COLLECT.

This opcode does the following:

- Sets the PIN_FLD_RESULT and PIN_FLD_DESCR values returned in the output flist.
- Specifies the payment events (for example, payment fees, payment reversals, and write-off reversals) to be performed on the account in the PIN_FLD_EVENTS array.
- Based on the results of the credit card transaction, specifies the actions to be performed on the account by returning a PIN_FLD_ACTIVITIES array.

PCM_OP_PYMT_POL_COLLECT is called by PCM_OP_PYMT_COLLECT after the credit card has been charged.

PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_RESULT and PIN_FLD_DESCR values returned in the output flist. It also specifies the actions to be performed on the account based on the results of the credit card transaction by returning a PIN_FLD_ACTIVITIES array.

The default behavior of PCM_OP_PYMT_POL_COLLECT is determined by the PIN_FLD_RESULT field passed in on the input flist.

- If the result is successful, PIN_CHARGE_RES_PASS is passed in. Depending on the PIN_FLD_COMMAND passed in on the flist, PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_DESCR value as shown in Table 21-4:

Table 21-4 Input and Output Values

Input PIN_FLD_RESULT	Input PIN_FLD_COMMAND	Output PIN_FLD_RESULT	Output PIN_FLD_DESCR
PIN_CHARGE_RES_PASS	PIN_CHARGE_CMD_AUTH_ONLY	PIN_RESULT_PASS	Authorization successful
PIN_CHARGE_RES_PASS	PIN_CHARGE_CMD_CONDITION	PIN_RESULT_PASS	Authorization & deposit successful
PIN_CHARGE_RES_PASS	PIN_CHARGE_CMD_DEPOSIT	PIN_RESULT_PASS	Deposit successful
PIN_CHARGE_RES_PASS	PIN_CHARGE_CMD_REFUND	PIN_RESULT_PASS	Refund successful

PCM_OP_PYMT_POL_COLLECT then specifies actions to be performed on the account based on the PIN_FLD_COMMAND, the payment amount, and the pending receivable amount as shown in Table 21-5:

Table 21-5 PCM_OP_PYMT_POL_COLLECT Actions

Input PIN_FLD_COMMAND	Actions When Payment >= Pending Receivable	Action When Payment < Pending Receivable
PIN_CHARGE_CMD_AUTH_ONLY PIN_CHARGE_CMD_CONDITION PIN_CHARGE_CMD_REFUNDS	– Clear the pending receivable amount. – If the account status is currently set to inactive, change it to active. – Set the status flag value to PIN_STATUS_FLAG_DEBT.	Credit toward outstanding bill.
PIN_CHARGE_CMD_DEPOSIT	No action specified	No action specified

- If the result is unsuccessful, and PIN_CHARGE_RES_FAIL_CARD_BAD or PIN_CHARGE_RES_FAIL_DECL_HARD are passed in, PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_RESULT value and the PIN_FLD_DESCR description, and then specifies these actions as shown in Table 21-6:

Table 21-6 Input and Output Values

Input PIN_FLD_RESULT	Output PIN_FLD_RESULT	Output PIN_FLD_DESCR	Action
PIN_CHARGE_RES_FAIL_CARD_BAD PIN_CHARGE_RES_FAIL_DECL_HARD	PIN_RESULT_FAIL	Credit card operation declined	Set account status to inactive. Set status flag value to PIN_STATUS_FLAG_DEBT.

The PIN_FLD_PAYMENT_REASONS array in the input flist contains a PIN_FLD_PAYMENT_REASONS array, which contains information related to failed payments. These values are recorded in the FAILED_ACCOUNTS array of the **/process_audit/billing** object.

> **Note:**
>
> If a single payment is submitted for multiple bills, and fails, it is stored as multiple FAILED_ACCOUNTS arrays.

The remaining input PIN_FLD_RESULT values are implemented in the same way. PCM_OP_PYMT_POL_COLLECT sets the output PIN_FLD_RESULT value and the PIN_FLD_DESCR description, and then reads the item for the account to determine if there is an amount that is 30 days past due. If so, it specifies the following actions shown in Table 21-7:

Table 21-7 Input and Output Values

Input PIN_FLD_RESULT	Output PIN_FLD_RESULT	Output PIN_FLD_DESCR	Action
PIN_CHARGE_RES_FAIL_DECL_SOFT PIN_CHARGE_RES_FAIL_ADDR_AVS PIN_CHARGE_RES_FAIL_ADDR_LOC PIN_CHARGE_RES_FAIL_ADDR_ZIP PIN_CHARGE_RES_FAIL_NO_ANS PIN_CHARGE_RES_SRVC_UNAVAIL	PIN_RESULT_FAIL	Credit card operation declined	Set account status to inactive. Set status flag value to PIN_STATUS_FLAG_DEBT.

You can customize PCM_OP_PYMT_POL_COLLECT to specify any of these actions:

- **clear_pending**
- **set_status**
- **cease_billing**

For example, to discontinue billing an account after a determined period of inactivity, specify the **cease_billing** action. The default implementation does not specify the **cease_billing** action for any input PIN_FLD_RESULT values.

Processing Paymentech Address Validation Return Codes

Paymentech provides return codes when verifying customer addresses. To change how BRM responds to validation return codes, edit PCM_OP_PYMT_POL_VALIDATE source.

For example, by default a invalid address does not cause a validation failure. You can change the policy to fail validation if the customer's street address is wrong. For example, change the following code:

```
case PIN_CHARGE_RES_FAIL_ADDR_LOC:
   /* street address failure is acceptable */
   result = PIN_RESULT_PASS;
   descr = "street address not correct";
   break;
```

To this:

```
case PIN_CHARGE_RES_FAIL_ADDR_LOC:
   /* street address failure is acceptable */
   result = PIN_RESULT_FAIL;
   descr = "street address not correct";
   break;
```

PCM_OP_PYMT_POL_VALIDATE returns the result of validating a credit card transaction in the PIN_FLD_VENDOR_RESULTS field, including a description of that result. You can customize credit card validations based on the response from ACH by passing a PIN_FLD_VENDOR_RESULTS value in the input flist. For example, you can set the validation to pass or fail, or turn on logging based on the results from the ACH.

PCM_OP_PYMT_VALIDATE calls PCM_OP_PYMT_POL_VALIDATE during customer account creation to determine the success or failure of credit card validation.

You can change both the PIN_FLD_RESULT and PIN_FLD_DESCR values to modify BRM responses to validation results returned in PIN_FLD_VENDOR_RESULTS. For example, if your company does not want to proceed with a transaction when the result is PIN_CHARGE_RES_SRV_UNAVAIL, change the PIN_FLD_RESULT_PASS value to PIN_FLD_RESULT_FAIL and change the PIN_FLD_DESCR value to **"Service unavailable"**.

If the PIN_FLD_RESULT value passed in on the input flist is **NULL**, PCM_OP_PYMT_POL_VALIDATE does nothing. Otherwise, the default validation result depends on the PIN_FLD_RESULT value. See Table 21-3.

> **Note:**
>
> If you add custom result values to your system, do not assign them the following result codes, which are reserved by BRM: **0** - **17**, **777**, **888**, **999**, **1000** - **1017**, **1777**, and **1999**.

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

Processing AVS Validations for International Credit Cards

By default, BRM sends a customer's name, address, and ZIP code for validation when processing credit card charges. If you use the Address Verification System (AVS) to validate addresses, only credit cards with addresses in the United States and Canada pass validation.

To change how BRM handles credit card address verifications, do one of the following:

- Set the **cc_validate** flag in the CM's **pin.conf** file to **0**. This disables the address validation process by Paymentech for all credit cards, including United States and Canada credit cards.
- Set the **cc_validate** flag in the CM's **pin.conf** file to **1** and modify PCM_OP_PYMT_POL_VALIDATE to ignore all AVS failure response codes for other countries. This changes how BRM responds to Paymentech's validation return codes for countries other than the United States and Canada.

Processing Direct Debit Charges

To perform a batch of Paymentech direct debit transactions, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_DD.

PCM_OP_PYMT_CHARGE_DD supports all commands handled by PCM_OP_PYMT_CHARGE, except that it does not create a payment structure and handles transaction charges of $1 only.

PCM_OP_PYMT_CHARGE_DD is used for the Paymentech direct debit implementation shipped with BRM, and is available for you to use in creating a custom direct debit implementation for the bank or payment clearing house you choose.

> **Note:**
>
> Debit cards that require a personal identification number (PIN) are not supported.

About Paymentech Direct Debit Implementation

PCM_OP_PYMT_CHARGE_DDEBIT performs a debit card transaction. This opcode is used for the Paymentech direct debit implementation, and is available for creating a custom direct debit implementation for the bank or payment clearing house you choose. Unless the PCM_OPFLG_CALC_ONLY flag is specified, PCM_OP_PYMT_CHARGE_DDEBIT creates an **/event/billing/charge/ddebit** object for each operation. If an array of operations is specified, then more than one event object is created.

PCM_OP_PYMT_CHARGE_DDEBIT supports all commands handled by PCM_OP_PYMT_CHARGE.

> **Important:**
>
> Debit cards that require a personal identification number (PIN) are not supported.

> **Note:**
>
> Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example, Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

Creating a Custom Direct Debit Implementation

By default, PCM_OP_PYMT_CHARGE_DD returns direct debit payment information, a charge status, and a payment status to the calling opcode for updating respective events. Effectively this opcode performs a loopback operation that you must change before you can implement direct debit charging. It does not output transaction data for a direct debit clearinghouse. BRM users must create an application to extract the information from the database for a specific direct debit clearinghouse.

Processing a Batch of Direct Debit Charges

To perform a batch of Paymentech direct debit transactions, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_DD. The processing is performed on a per-batch basis; only one command and one payment method can exist in the same batch.

PCM_OP_PYMT_CHARGE_DD supports all commands handled by PCM_OP_PYMT_CHARGE, except that it does not create a payment structure and handles transaction charges of $1 only.

The input flist contains a PIN_FLD_SESSION_OBJ field, which defines the session in which the event occurred: either a payment batch event or a refund batch event (**/event/billing/batch/payment** or **/event/billing/batch/refund**).

Unless the PCM_OPFLG_CALC_ONLY flag is specified, PCM_OP_PYMT_CHARGE_DD creates an **/event/billing/charge/dd** object for each operation. If an array of operations was specified, then more than one event object is created. The event objects are created even if the direct debit operations cannot be performed.

> **Note:**
>
> Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example, Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

The set of Paymentech return codes handled by BRM is listed in the *BRM_home***/sys/dm_fusa/fusa_codes** file. These codes can be modified.

Checking the Results of BRM-Initiated Batch Payment Operations

To check the results of batch payment operations, set the PIN_FLD_COMMAND value in the PCM_OP_PYMT_COLLECT input flist to PIN_CHARGE_CMD_RECOVER_PAYMENT. This causes PCM_OP_PYMT_CHARGE to call PCM_OP_PYMT_RECOVER.

PCM_OP_PYMT_RECOVER posts the results of charges for which no information was returned.

PCM_OP_PYMT_RECOVER calls the following opcodes:

- To check the results of a batch of credit card charges, PCM_OP_PYMT_RECOVER_CC calls PCM_OP_PYMT_RECOVER_CC. This opcode posts results of credit card charges for which no information was returned.

  PCM_OP_PYMT_RECOVER_CC is specific to the Paymentech DM.

- To check the results of a batch of direct debit charges, PCM_OP_PYMT_RECOVER_CC calls PCM_OP_PYMT_RECOVER_DD. This opcode posts results of direct debit charges for which no information was returned. The results are passed back and used for transaction reconciliation.

  PCM_OP_PYMT_RECOVER_DD is specific to the Paymentech DM.

Validating Credit Card and Direct Debit Transactions

To validate credit card and direct debit transactions, use PCM_OP_PYMT_VALIDATE. This opcode is called by PCM_OP_CUST_PREP_CUSTOMER and PCM_OP_CUST_POL_VALID_PAYINFO during account creation.

PCM_OP_PYMT_VALIDATE calls PCM_OP_PYMT_POL_VALIDATE to determine the success or failure of a BRM-initiated transaction validation.

PCM_OP_PYMT_VALIDATE reads the **/config/payment** storable class to determine the transaction type and the opcode to call, and then calls the appropriate opcode to validate the transaction.

- Credit card transactions: PCM_OP_PYMT_VALIDATE_CC.

  PCM_OP_PYMT_VALIDATE_CC performs a batch of online credit card validations and applies the validation policy to the results.

- Direct debit transactions: PCM_OP_PYMT_VALIDATE_DD.

  PCM_OP_PYMT_VALIDATE_DD performs a batch of online direct debit validations and applies the validation policy to the results. PCM_OP_PYMT_VALIDATE_DD calls the appropriate DM to process validations, then returns the results to the Internet.

For both opcodes, the input flist contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified in each operation, so a single batch can contain different commands. The PIN_FLD_SESSION_OBJ in the input flist is either **/event/billing/batch/refund** or **/event/billing/batch/payment**, depending on the batch type: payment or refund.

In addition, PCM_OP_CUST_PREP_CUSTOMER calls PCM_OP_PYMT_POL_SPEC_VALIDATE to determine whether a customer's payment information needs to be validated during account creation. If validation is required, PCM_OP_CUST_PREP_CUSTOMER calls PCM_OP_PYMT_VALIDATE to perform the operation. If validation fails, the results are transformed to be consistent with the results used to describe account creation failure results and are then returned on the output flist.

When validating a credit card at account creation, BRM needs an account to validate the card with. By default, this is the root account. You cannot store this information with each account because the credit card validation is done before the account is created.

Processing Credit Card Information during Account Creation

During the account creation process, PCM_OP_CUST_COMMIT_CUSTOMER passes credit card information to PCM_OP_PYMT_VALIDATE and PCM_OP_PYMT_COLLECT, which collect any credit card payments charged at account creation and validate the credit card information returned by the payment processor.

- PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_POL_SPEC_COLLECT, which passes the bill unit associated with the payment and returns a list of open items to be paid in the PIN_FLD_ITEMS array.
- calls PCM_OP_PYMT_GET_ACH_INFO to retrieve the Automated Clearing House (ACH) information. PCM_OP_PYMT_POL_SPEC_COLLECT allows you to determine whether to charge the customer immediately for all or part of the current account balances during account creation.

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

If the **CCcollect** business parameter is set to **enabled**, during account creation for credit card accounts, the total due amount for the account is charged immediately and the payment is allocated immediately to all open bill items. Therefore, after the account is created, it will have no pending amount due and no unapplied payments.

Storing Card Credentials for Future Transactions

To store a customer's credit card credentials for future transactions, use PCM_OP_PYMT_COLLECT.

This opcode does the following:

1. Receives the PIN_FLD_TRANSACTIONS array from the following opcodes when a VISA, MasterCard, Diners, Discover, JCB, or American Express card is registered for payment:
   - PCM_OP_CUST_COMMIT_CUSTOMER: This opcode passes the PIN_FLD_TRANSACTIONS array as input when you register a new customer with **Credit Card** as the default payment method and either credit card tokenization is enabled or the **cc_validate** flag in the CM or the **CCcollect** business parameter is enabled.
   - PCM_OP_CUST_SET_PAYINFO: This opcode passes the PIN_FLD_TRANSACTIONS array as input when you set **Credit Card** as the default payment method for an account.
   - PCM_OP_CUST_CREATE_PAYINFO: This opcode passes the PIN_FLD_TRANSACTIONS array as input when you add a new credit card and set that as the default payment method for the account.
2. Accepts the card information in the PIN_FLD_TRANSACTIONS array, adds missing information as required, and then passes the information as input to the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode.

   > **Note:**
   >
   > You can add, update, or remove the card information in the PIN_FLD_TRANSACTIONS array by customizing the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode.

3. Receives the updated card information and the PIN_FLD_CHARGES array from the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode.
4. Sends the card information in the PIN_FLD_TRANSACTIONS array to the Paymentech DM by calling the PCM_OP_PYMT_CHARGE or PCM_OP_PYMT_CHARGE_CC opcode.

   The Paymentech DM appends the required records based on the information received and sends the transactions to Paymentech. If the transaction is successful, the Paymentech DM retrieves the TXID from the Paymentech response and passes it to the PCM_OP_PYMT_COLLECT opcode.
5. Accepts the TXID received from the Paymentech DM and stores it in the PIN_FLD_TRANSACTIONS array in the **/payinfo/cc** object for future transactions.

Purging Card Credentials

When an account's default payment method is **Credit Card**, PCM_OP_PYMT_COLLECT stores the card credentials for each bill unit in the PIN_FLD_TRANSACTIONS array of the **/payinfo/cc** object.

If the default payment method is changed to any other method or card, the PIN_FLD_TRANSACTIONS array in the **/payinfo/cc** object is automatically purged from the database if the account payments are not in cardholder-initiated installments. If the payments are in cardholder-initiated installments, you must delete the **/payinfo/cc** object's PIN_FLD_TRANSACTIONS array manually after the payment is made for the last installment.

About Credit Card Payment Confirmation Numbers

When a credit card payment completes successfully, BRM returns a confirmation number that the customer can use later to identify the payment. BRM uses the payment item number as the confirmation number.

PCM_OP_PYMT_COLLECT returns the confirmation number in the PIN_FLD_ITEM_NO output flist field of the PIN_FLD_RESULTS array.

Externally Initiated Payment Processing

For externally initiated payment processing, the normal flow of PCM_OP_PYMT_COLLECT is as follows:

1. Calls PCM_OP_PYMT_VALIDATE_PAYMENT to determine the status of the payment records. See "Validating Payments" for more information.
2. Calls PCM_OP_PYMT_POL_PRE_COLLECT, to perform policy checks before the charge, payment, or refund occurs after allocating the PIN_FLD_CHARGE array elements to open items.
3. Calls PCM_OP_PYMT_SELECT_ITEMS to identify the items to which the payments in the batch are applied. See "How BRM Selects the Items to Which Payments Are Applied".

   If necessary, PCM_OP_PYMT_COLLECT then calls PCM_OP_PYMT_POL_OVER_PAYMENT and PCM_OP_PYMT_POL_UNDER_PAYMENT, to allocate overpayment and underpayment of funds, respectively.
4. Calls the following opcodes:
   - Calls PCM_OP_BLL_RCV_PAYMENT to create the **/event/billing/payment/pay_type** object.

     If the payment status is marked as a failed unconfirmed payment, PCM_OP_BILL_REVERSE_PAYMENT reverses the unconfirmed successful payments that have a value of PIN_PYMT_FAILED in the PIN_FLD_STATUS field. The PIN_FLD_AMOUNT value in the input flist CHARGES array is set to the value of PIN_FLD_AMOUNT_ORIGINAL_PAYMENT, and the value of PIN_FLD_AMOUNT is set to **0**.
   - Calls PCM_OP_PYMT_ITEM_TRANSFER to allocate the payment to open items.
   - Calls PCM_OP_PYMT_APPLY_FEE to record failed payments and apply payment fees. If a payment fee is applied, the POID of the payment fee event is added to the PCM_OP_PYMT_APPLY_FEE output flist.

- If the **cease_billing** action was received, calls PCM_OP_CUST_SET_BILLINFO to change the status of the bill unit.

5. If mandated by the policy FM, calls PCM_OP_PYMT_POL_COLLECT to perform the following actions listed in Table 21-8, which are based on the payment result:

Table 21-8 PCM_OP_PYMT_POL_COLLECT Policy Opcode Actions

Action	Description
clear_pending	Apply the payment to reduce the pending receivable of the bill unit by the amount specified.
set_status	Change the bill unit's status to that given, using the reasons indicated by the flags.
issue_refund	Apply a refund by crediting the bill unit with the refund amount specified.
cease_billing	Discontinue billing an account after a determined period of inactivity by marking the given bill unit as no longer billed.

6. Creates the final batch.

Receiving Payments

When payments are received in BRM, they are processed by PCM_OP_PYMT_COLLECT. To record the payments, PCM_OP_PYMT_COLLECT calls PCM_OP_BILL_RCV_PAYMENT. PCM_OP_BILL_RCV_PAYMENT records the payment items and events and updates account balances by calling PCM_OP_BILL_ITEM_TRANSFER. If there is excess money after paying the item list off, PCM_OP_BILL_RCV_PAYMENT adds that amount to the sub-balance as a credit. If there is no sub-balance, one is created.

- When posting a suspended payment, PCM_OP_BILL_RCV_PAYMENT stores the reason code, action owner code, and original reason code for the payment in the payment event's PIN_FLD_EVENT_MISC_DETAILS array. The reason codes are passed in from the PCM_OP_PYMT_COLLECT input flist's PYMT_REASONS array.

  The PIN_FLD_EVENT_MISC_DETAILS array uses specific element IDs as follows:

  – **Array Element 0**: Stores the reason code used to determine the G/L ID of the payment being moved to the payment suspense account.

  – **Array Element 1**: Stores the action owner code for suspended or failed payments.

  – **Array Element 2**: Stores the original reason code for failed payments that BRM suspended. This ensures that the reason initially associated with the financial failure is not lost if BRM places the payment in suspense. Element 2 is used only for Payment Suspense Manager; if payment suspense is not enabled, only element 0 is present.

  When the failed payment leaves suspense and PCM_OP_BILL_RCV_PAYMENT creates new payment objects to record the corrected payment, Elements 0 and 1 are no longer needed. The object does not contain the elements 0 and 1 recorded for suspense, and Element 2 becomes the new Element 0.

- When processing multiple voucher top-ups that include noncurrency balances, PCM_OP_BILL_RCV_PAYMENT retrieves the noncurrency balance impacts from the PIN_FLD_TOPUP_RESOURCE_INFO substruct in the PCM_OP_PYMT_COLLECT input flist and passes them to PCM_OP_ACT_USAGE so that they are recorded in the corresponding **/event/billing/payment/voucher** object. See "Processing Top-Ups" for more information.

PCM_OP_BILL_RCV_PAYMENT uses the PIN_FLD_SESSION_OBJ field in the input flist to reference the type of session in which the event occurred: either **/event/billing/batch/refund** or **/event/billing/batch/payment**, depending on the batch.

Validating Payments

Before payments can be processed and posted by BRM, they must pass validation. Payment validation is initiated automatically, through the payment gateway, or manually, by a payment clerk. When a payment arrives in BRM, PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_VALIDATE_PAYMENT to perform the validation. BRM validates the payment information in the input flist and, in the output flist, indicates whether the opcode successfully performed the validation.

The PIN_FLD_STATUS value returned by PCM_OP_PYMT_VALIDATE_PAYMENT indicates whether the payment arrived in BRM as *successful* or *failed for financial reasons*. If you have the Payment Suspense Manager functionality installed, payments can also arrive as *suspended*.

- If the payment is *successful* and passes the validation process, the PIN_FLD_STATUS value is PIN_PYMT_SUCCESS, and BRM posts the payment to the account.
- If the payment meets the validation criteria, but fails for financial reasons, the PIN_FLD_STATUS value is PIN_PYMT_FAILED, and BRM posts the payment to the account as a failed payment.
- If the payment status is marked for suspense, the PIN_FLD_STATUS value is PIN_PYMT_SUSPENSE, and PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_POL_SUSPEND_PAYMENT to process the payment. PCM_OP_PYMT_POL_SUSPEND_PAYMENT places a payment in suspense by enriching the flist so that the payment can be directed to the payment suspense account. It then calls the opcodes that post the payment to this account and create objects to record the suspended payment. The event object contains all information about the payment and its suspense, including the account number, bill number, transaction ID, and any associated reason codes or action owner codes. This information can be used to investigate why the payment failed the validation process and who is responsible for resolving the problem.

If an account payment is made to an account with multiple bill units, PCM_OP_PYMT_POL_VALIDATE_PAYMENT validates that the payment is an account payment and the account has multiple bill units. If the payment is successful and passes the validation process, it adds the reason ID, PIN_REASON_ID_MBI_DISTRIBUTION_REQD to the PIN_FLD_PAYMENT_REASONS array in the output flist. This reason ID helps in later payment processing by distinguishing a normal payment from an account payment made to an account with multiple bill units.

If the payment fails the validation, BRM informs you that the payment cannot be posted. For example, if a payment specifies no account number and no bill number, BRM cannot post the payment. You must create an exception batch to handle payments that fall into this category.

About the Default Payment Validation Process

PCM_OP_PYMT_POL_VALIDATE_PAYMENT enables you to customize how to validate payments to determine whether they can be successfully posted or whether a failed, unconfirmed payment needs reversal.

This opcode also identifies if the account payment is made to accounts with multiple bill units. During validation, this opcode tries to find any missing data needed to process payments. If automatic write-off reversals are enabled, this opcode also determines whether BRM should perform a write-off reversal.

PCM_OP_PYMT_POL_VALIDATE_PAYMENT opcode processes payments in three phases:

1. **Payment suspense phase**: Receives a list of payments from PCM_OP_PYMT_VALIDATE_PAYMENT and checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled. If so, it checks the payments to determine whether any payments must be suspended and updates the PIN_FLD_STATUS field accordingly.
2. **Failed unconfirmed payment phase**: For all payments with a PIN_FLD_STATUS in the failed range, uses the payment method value to determine whether a failed payment has an associated unconfirmed successful payment or if it is a failed confirmed payment.

   The input flist contains the POID of the original payment item, the transaction ID, and the amount of the original payment to properly handle unconfirmed failed payments.
3. **Write-off reversal phase**: Checks the value of the **AutoWriteOffReversal** business parameter to determine whether automatic write-off reversals are enabled. If so, it checks the **WriteOffLevel** business parameter to determine whether the payment is for an account, bill, or bill item that has been written off. If so, it sets PIN_FLD_STATUS accordingly.

Payment Suspense Phase

In this phase, PCM_OP_PYMT_POL_VALIDATE_PAYMENT first checks the PIN_FLD_STATUS field to determine whether the payment has a status in the suspense range, indicating that the payment has already been marked for suspense. In this case, the opcode passes the output flist and associated status back to PCM_OP_PYMT_VALIDATE_PAYMENT. BRM will then use PCM_OP_PYMT_COLLECT to direct the payment to the payment suspense account.

If the payment is not already marked for suspense, PCM_OP_PYMT_POL_VALIDATE_PAYMENT does the following:

1. Validates the account number specified by the PIN_FLD_ACCOUNT_NO or PIN_FLD_ACCOUNT_OBJ field in the input flist or searches for the corresponding account POID.
2. Validates the bill number specified by the PIN_FLD_BILL_NO field in the input flist and searches for the corresponding bill POID, bill unit POID, and account POID.
3. If neither a bill POID nor a bill number was submitted with the payment, BRM uses the bill amount to find the bill.

   If the account and bill numbers supplied in the input flist are both invalid or PCM_OP_PYMT_POL_VALIDATE_PAYMENT cannot find the account or bill POIDs (or bill amount), it marks the payment for suspense. It also compares the account POID from the ***/bill*** object with the account POID found in step 1. If they do not match, the opcode marks the payment for suspense.
4. Checks the PIN_FLD_STATUS field in the ***/account*** object to determine whether the account is closed. If the account is closed, PCM_OP_PYMT_POL_VALIDATE_PAYMENT marks the payment for suspense.
5. Checks any custom validation criteria and marks the payment for suspense if appropriate.
6. Writes the status of the payment in the PIN_FLD_STATUS field, and includes this field in the output flist. If the payment must be suspended, it sets this field to one of the values in the suspense range, as appropriate.

> **Note:**
>
> - When an **/event/billing** object is created for a suspended payment, it stores the original reason code associated with a failed payment that has been flagged for suspense. This ensures that the reason initially associated with the failed payment is not lost if BRM places the payment in suspense.
> - When the **/event/billing/payment** object is created, it stores the original account number provided for the payment being suspended, the original bill number, and the original transaction ID.

Failed Unconfirmed Payments Phase

In this phase, the opcode considers only payments whose status is marked as failed: those whose PIN_FLD_STATUS value is in the financially failed range. These payments are ones that the opcode was able to validate, but were marked by the payment processor as failing for financial reasons. In default implementations, the opcode requires the transaction ID and result of each payment to prepare the failed unconfirmed payments for reversal.

If the failed payment is an unconfirmed payment, PCM_OP_PYMT_VALIDATE_PAYMENT:

1. Searches the **/event/billing/payment** object for an unconfirmed payment with the transaction ID passed in with the failed payment.
2. Does one of the following:
   - If an unconfirmed payment is found, sets PIN_FLD_RESULT to PIN_PAY_TYPE_SUCCESS.
   - If the transaction ID of the unconfirmed payment is not found, it checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled.

     If so, it sets PIN_FLD_STATUS to PIN_FLD_FAILED_SUSPENSE and BRM posts the payment to the payment suspense account.

     If not, it sets PIN_FLD_RESULT to PIN_FLD_PAYMENT_RESULT_FAIL, and a reversal does not occur. The subsequent steps do not occur and manual allocation is required.
3. Loads the following unconfirmed payment information from the **/event/billing/payment** storable class into the PIN_FLD_FAILED_PAYMENT_FEE substruct in the PIN_FLD_EXTENDED_INFO substruct of the output flist:
   - Payment channel ID
   - Payment method
   - Transaction ID
   - Original payment amount
   - Customer segment

   > **Note:**
   >
   > For unconfirmed payments, the customer segment value is also retrieved from PIN_FLIST_CUSTOMER_SEGMENT_LIST in the input flist of this policy.

4. Passes the POID of the successful unconfirmed payment in the output flist to PCM_OP_BILL_REVERSE_PAYMENT so it can be reversed.

The output flist sends the array of reversal events and tax events (if created) that were passed in by PCM_OP_AR_WRITEOFF_REVERSAL.

Write-off Reversal Phase

In this phase, PCM_OP_PYMT_VALIDATE_PAYMENT considers only payments whose status is marked as successful: those whose PIN_FLD_STATUS value is in the successful range. The opcode determines which of these payments is for an account, bill, or bill item that has been written off. It performs the following operations:

- Checks the value of the **AutoWriteOffReversal** business parameter to determine whether automatic write-off reversal functionality for payment processing is enabled.
- If this is enabled, checks the **/profile/writeoff** object to verify that the write-off flag is set for the account.
- If both checks are successful, sets the PIN_FLD_STATUS field in the output flist to PIN_PYMT_WRITEOFF_SUCCESS.

About Payment Validation Flags

Flags are not used directly by PCM_OP_PYMT_POL_VALIDATE_PAYMENT. They are passed in from PCM_OP_PYMT_COLLECT for PCM_OP_PYMT_SELECT_ITEMS.

Configuring Unconfirmed Payment Processing

BRM requires acknowledgment from a bank or payment processor before posting BRM-initiated payments. In some cases, the response from the bank or payment processor does not occur immediately with the request for funds. In that case, you can allow BRM to post unconfirmed payments.

To avoid the possible delay in posting payments, you can configure a new payment Data Manager (DM) to post payments immediately, before the funds are confirmed by the bank or payment processor. The DM requires an input flist of payments from BRM and must return the results to BRM in the output flist.

> **Note:**
>
> Only credit card and debit card payment methods can be posted before they are confirmed.

If the bank or payment processor later sends a failure notification for a BRM-initiated payment (for example, due to insufficient funds or an expired credit card), BRM reverses the initially-successful payments and posts the failed payments.

Failed unconfirmed payments can be submitted to BRM by using a payment gateway, or by using another third-party payment application.

Failed unconfirmed payments are recorded as **/event/billing/payment/failed** events, and the reversals of the initially-successful payments are recorded as **/event/billing/reverse/***pay_type* events.

When a unconfirmed payment fails, the failed payment is recorded in BRM with a balance impact of **0**, and the successful payment amount is re-applied to the account balance.

Failed unconfirmed payments are processed by the PCM_OP_PYMT_COLLECT opcode. The interface you use to load failed payments into the BRM database must be configured to send the following information with each failed payment to be validated in BRM:

- Transaction ID
- Failed payment status
- Failure reason ID

The transaction ID of the failed payment is compared against the transaction ID in the **/event/billing/payment/***pay_type* object of the initial unconfirmed payment that was posted successfully in BRM.

If the transaction IDs are the same, the original payment is reversed and the failed payment is recorded. If the transaction ID is missing or incorrect, or the successful payment cannot be located in the database, the PCM_OP_PYMT_VALIDATE_PAYMENT opcode returns a value of **PIN_PYMT_FAILED** in the PIN_FLD_RESULT field, and an error is displayed. The payment is not reversed and the failed payment is not allocated. You must manually fix the transaction ID to resubmit the payment. Or, you can configure BRM to identify the original payment by using other payment attributes.

When a failed unconfirmed payment is received, the original successful payment is identified by using its transaction ID.

The PIN_FLD_PAYMENT_TRANS_ID field in the **/event/billing/payment/failed** event must be equal to the PIN_FLD_TRANS_ID of the successful unconfirmed payment item. If so, the unconfirmed payment is reversed and the failed payment posted to the proper account.

If the payment processor is not able to send a transaction ID with each payment, or if the transaction ID is not a reliable means of identifying a payment, you can configure PCM_OP_PYMT_POL_VALIDATE_PAYMENT to find the original unconfirmed payment by using other payment properties. For example, you can use a combination of the payment amount, account number, and invoice number.

When the payment is received, BRM compares these values in the failed payment with those of the original unconfirmed payment, and if the values match, it will reverse the unconfirmed payment and post the failed payment.

> **Note:**
>
> The result of the validation is the POID of the unconfirmed successful payment item that was recorded in BRM. If the item is not available, the reversal cannot occur. By default, this policy opcode retrieves the item by finding the payment event with corresponding transaction ID.

Handling Overpayments and Underpayments

If a customer pays too much or too little, your Oracle Communications Billing and Revenue Management (BRM) business policies may require payment allocation.

- For underpayments, choose which bills or items to allocate the payment to.
- For overpayments, pay all items and generate a credit balance.

By default, BRM requires overpayments to be allocated.

To change the default BRM behavior for overpayments and underpayments, customize the PCM_OP_PYMT_POL_UNDER_PAYMENT or PCM_OP_PYMT_POL_OVER_PAYMENT policy opcodes.

If the money received is more or less than the sum of the total due of all the open items selected by PCM_OP_PYMT_SELECT_ITEMS, PCM_OP_PYMT_SELECT_ITEMS calls PCM_OP_PYMT_POL_OVER_PAYMENT or PCM_OP_PYMT_POL_UNDER_PAYMENT, respectively. PCM_OP_PYMT_SELECT_ITEMS does *not* call these policy opcodes if:

- The amount is equal to the sum of the total due.
- The flag PIN_BILLFLG_SELECT_FINAL is passed in.
- The flag PIN_BILLFLG_DEFER_ALLOCATION is passed in.

By default, PCM_OP_PYMT_POL_OVER_PAYMENT returns the amount overpaid on the output flist. The excess amount remains in the bill unit (**/billinfo** object) specified on the input flist (or the default bill unit if none was specified) until they are manually redistributed by using Billing Care or Customer Center. You can customize PCM_OP_PYMT_POL_OVER_PAYMENT to perform as a hook for an application that would search for and settle all overpaid payment items.

By default, PCM_OP_PYMT_POL_UNDER_PAYMENT pays the billed items in the order they are listed on the input flist (**item[0]** first, then **item[1]**, **item [2]**, and so on). It then returns the items paid on the output flist. Items that are partially paid are returned with a new amount due. Items not paid are not returned.

Calculating Payment Collection Dates

By default, BRM-initiated payments, such as payments made by credit card or direct debit, are collected on the date that bills are finalized. Alternatively, you can configure BRM to collect a BRM-initiated payment on the date a bill is *due* or on a specified number of days *before* the bill is due.

To support configurable payment collection dates, PCM_OP_BILL_POL_CALC_PYMT_DUE_T calculates a bill's payment collection date after calculating its due date.

> **Note:**
>
> Although configurable payment collection dates are used only for BRM-initiated payments, they are calculated and stored for bills associated with *all* payment methods.

To calculate payment collection dates, PCM_OP_BILL_POL_CALC_PYMT_DUE_T performs these tasks:

1. Finds the **/payinfo** object that is linked to the bill unit with which the bill is associated.
2. Reads the value of the PIN_FLD_PAYMENT_OFFSET field in the **/payinfo** object.
3. Does one of the following:
   - If the value is **-1**, sets the payment collection date to the date the bill is finalized.
   - If the value is **0**, sets the payment collection date to the date the bill is due.

- If the value is any positive integer (x), sets the payment collection date to x days *before* the bill is due.

> **Note:**
>
> If x makes the payment collection date earlier than the date the bill is finalized, PCM_OP_BILL_POL_CALC_PYMT_DUE_T uses the finalization date instead.

4. Stores the payment collection date in the PIN_FLD_COLLECTION_DATE field of the **/billinfo** object.

> **Note:**
>
> By default, the **pin_collect** utility collects BRM-initiated payments for all bills associated with **/billinfo** objects whose PIN_FLD_COLLECTION_DATE is the day the utility is run or the day before the utility is run.

How BRM Selects the Items to Which Payments Are Applied

PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_SELECT_ITEMS to identify the items to which the payment is applied.

When an account payment is made to an account having multiple bill units, this opcode processes more than one bill unit to get the item distribution corresponding to each bill unit.

In case of overpayment to an account, PCM_OP_PYMT_SELECT_ITEMS contains more than two PIN_FLD_BILLINFO arrays for the default bill unit. PCM_OP_PYMT_SELECT_ITEMS does not perform an item distribution for the second PIN_FLD_BILLINFO array for the default bill unit and sets the select status as PIN_SELECT_STATUS_OVER_PAYMENT. This restriction prevents the opcode from doing item distribution twice in two different PIN_FLD_BILLINFO arrays.

PCM_OP_PYMT_SELECT_ITEMS does the following:

1. Selects open items based on the input fields and the accounting type of the account.
2. If the amount passed in on the input flist is not in the primary account currency, PCM_OP_PYMT_SELECT_ITEMS attempts to convert it to the primary account currency.
3. One of the following actions is taken:
   - If called by PCM_OP_PYMT_COLLECT, PCM_OP_PYMT_SELECT_ITEMS returns the PIN_FLD_AMOUNT and the list of selected items.
   - If it is an overpayment, it calls PCM_OP_PYMT_POL_OVER_PAYMENT. By default, PCM_OP_PYMT_POL_OVER_PAYMENT returns the amount overpaid on the output flist.

     If it is an underpayment, it calls PCM_OP_PYMT_POL_UNDER_PAYMENT.

     By default, PCM_OP_PYMT_POL_UNDER_PAYMENT pays the billed items in the order they are listed on the input flist (**item[0]** first, then **item[1]**, **item [2]**, etc.). It then returns the items paid on the output flist. Items that are partially paid are returned with a new amount due. Items not paid are not returned.

4. Returns the list of selected items.

Items are selected by PCM_OP_PYMT_SELECT_ITEMS based on the fields in the input flist.

- The contents of the PIN_FLD_BILLS array is examined. For BRM-initiated payments, the contents of the PIN_FLD_BILLINFO array is also examined.
  - The PIN_FLD_BILLS array indicates which **/bill** object PCM_OP_PYMT_SELECT_ITEMS is collecting or paying off. If one or more bills are included, all items belonging to those bills are selected. If none is specified, the default bill unit is used to apply the payment.
  - The PIN_FLD_BILLINFO array specifies the bill unit to select items for. If none is specified, the items are retrieved based on the bills in the PIN_FLD_BILLS array. If *neither* a bill unit POID nor a bill is included, the items are selected from the account's default bill unit. This is the bill unit that contains the default balance group for the account.

> **Note:**
>
> For account payments to accounts with multiple bill units, there are multiple PIN_FLD_BILLINFO arrays corresponding to each bill unit that contains the bill unit–level distribution.

- If the accounting type is PIN_ACTG_TYPE_OPEN_ITEMS or if the accounting type is PIN_FLD_ACTG_TYPE_BALANCE_FORWARD and the PIN_FLD_BILLS array is passed in, there are two options for items to be eligible for the selection criteria:
  - If the PIN_FLD_INCLUDE_CHILDREN field is not specified or is specified and set to **1**, items that belong to the A/R bill unit (including nonpaying child bill items) are selected.
  - If the PIN_FLD_INCLUDE_CHILDREN field is specified and set to **0**, items that belong only to the specified bill are selected.

> **Note:**
>
> PIN_FLD_INCLUDE_CHILDREN applies only when the PIN_FLD_BILLS array is specified.

- If PIN_FLD_ACTG_TYPE is PIN_FLD_ACTG_TYPE_BALANCE_FORWARD and PIN_FLD_BILLS is not specified, PCM_OP_PYMT_SELECT_ITEMS selects all the open items for this bill unit and sums up the amount due from all the open items selected.
- The PIN_FLD_AMOUNT field determines whether the overpayment or underpayment policy opcodes are called. If PIN_FLD_AMOUNT is not specified, the payment amount is based on the charges of the bill unit's open items.

  If the PIN_FLD_AMOUNT field is passed:
  - The payment is not allocated to other open items and the deferred allocation flag is set.
  - The payment must be allocated manually.
- If the PIN_FLD_BILLINFO array contains bill unit level payment distribution, PCM_OP_PYMT_SELECT_ITEMS finds out the item distribution for the selected bill units.

Allocating Account Payments to Multiple Bill Units

If an account payment is made to an account having multiple bill units, you can allocate the payment to multiple bill units of the account.

> **Note:**
>
> The Payment Suspense Management feature must be enabled in your BRM system for you to allocate account payments to multiple bill units.

To allocate an account payment to multiple bill units of an account, BRM calls PCM_OP_PYMT_COLLECT. PCM_OP_PYMT_COLLECT performs the following operations:

1. Opens a transaction.
2. Calls PCM_OP_PYMT_VALIDATE_PAYMENT to validate the payment.

   PCM_OP_PYMT_VALIDATE_PAYMENT invokes PCM_OP_PYMT_POL_VALIDATE_PAYMENT to validate the payment. This opcode:

   a. Checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled.

   b. Checks that the payment is an account payment and that the account has multiple bill units.

   c. If the payment is successful and passes the validation process, adds the reason ID, PIN_REASON_ID_MBI_DISTRIBUTION_REQD to the PIN_FLD_PAYMENT_REASONS array in the output flist.

   > **Note:**
   >
   > For failed or suspended payments, PCM_OP_PYMT_POL_VALIDATE_PAYMENT does not add the PIN_REASON_ID_MBI_DISTRIBUTION_REQD reason ID.

3. Calls PCM_OP_PYMT_MBI_DISTRIBUTE to distribute the payment to multiple bill units. PCM_OP_PYMT_MBI_DISTRIBUTE invokes PCM_OP_PYMT_POL_MBI_DISTRIBUTE if the following conditions are true:

   - PCM_OP_PYMT_POL_VALIDATE_PAYMENT returns the PIN_REASON_ID_MBI_DISTRIBUTION_REQD reason ID.
   - The PIN_FLD_SELECT_STATUS field in the PCM_OP_PYMT_COLLECT input flist is *not* PIN_SELECT_STATUS_MBI_DISTRUBUTED.

   PCM_OP_PYMT_MBI_ITEM_SEARCH gets all the items of the bill units in a tree view. The bills are displayed under their corresponding bill units, and the items for a bill are displayed under their corresponding bill. This opcode calls PCM_OP_PYMT_ITEM_SEARCH.

   PCM_OP_PYMT_POL_MBI_DISTRIBUTE distributes the payment according to the default distribution logic. You can customize how payments are distributed by using PCM_OP_PYMT_POL_MBI_DISTRIBUTE. This opcode searches for all the open **/bill**

objects of all the **/billinfo** objects of the specified **/account** object, sorted by the bill due date.

PCM_OP_PYMT_POL_MBI_DISTRIBUTE returns the PIN_FLD_BILLINFO array that contains an array of PIN_FLD_BILLS having the distributed payment amount for each bill. The PIN_FLD_BILLINFO array is added to the PCM_OP_PYMT_MBI_DISTRIBUTE output flist, which is passed to PCM_OP_PYMT_SELECT_ITEMS to get item distribution.

4. Calls PCM_OP_PYMT_SELECT_ITEMS for item payment distribution for each bill unit.
5. If the reason ID in the PCM_OP_PYMT_SELECT_ITEMS output flist is PIN_REASON_ID_MBI_DISTRIBUTION_REQD, PCM_OP_PYMT_COLLECT detaches the distribution part from the output flist to use later while recycling payment.
6. If the reason ID in the PCM_OP_PYMT_SELECT_ITEMS output flist is PIN_REASON_ID_MBI_DISTRIBUTION_REQD and the PIN_FLD_STATUS value is successful (value in the range of **0** to **14**), PCM_OP_PYMT_COLLECT changes the status to PIN_PYMT_SUSPENSE to suspend the payment before calling PCM_OP_PYMT_POL_PRE_COLLECT.
7. Calls PCM_OP_PYMT_POL_PRE_COLLECT to perform policy checks before the payment occurs.
8. Calls PCM_OP_PYMT_POL_SUSPEND_PAYMENT to get the suspense account.
9. Calls PCM_OP_BILL_RCV_PAYMENT to record the payment and create the payment item.
10. Prepares the PCM_OP_PYMT_RECYCLE_PAYMENT input flist by using the distribution detached from PCM_OP_PYMT_SELECT_ITEMS.
11. Calls PCM_OP_PYMT_RECYCLE_PAYMENT to distribute the suspended payment to multiple bill units.
12. Prepares the PCM_OP_PYMT_COLLECT output flist such that the output flist contains all the payment events created as a result of the recycle payment.

Default payment distribution follows these rules:

- Bills having older due dates receive the payment amount first.
- If all bills have the same due date, the bills with the higher due amounts are considered first for payment distribution.
- In case of overpayment, the excess payment amount remains unallocated to the default bill unit of the account.
- In case of underpayment, bills with later due dates or low due amounts do not get any payment amount.
- For hierarchical accounts, the bills for the parent are considered first.

> **Note:**
>
> By default, PCM_OP_PYMT_POL_MBI_DISTRIBUTE provides bill distribution. So, BRM considers only the open bill items for payment distribution. However, you can update this opcode to return bill-unit payment distribution. If bill-unit distribution is passed to PCM_OP_PYMT_SELECT_ITEMS, payment considers all the open items, even if an open item is a bill item or an A/R item.

Allocating Externally Initiated Payments by Due Amount

When allocating an externally initiated payment, such as a payment made by check or cash, that is associated with a valid account number but not a valid bill POID, BRM uses PCM_OP_PYMT_POL_VALIDATE_PAYMENT to find the appropriate bill as follows:

1. If the element associated with the payment in the PIN_FLD_CHARGES array of PCM_OP_PYMT_POL_VALIDATE_PAYMENT input flist has a bills array (PIN_FLD_BILLS) and the array contains a bill number (PIN_FLD_BILL_NO), the opcode searches for the bill POID associated with the bill number.
2. If PCM_OP_PYMT_POL_VALIDATE_PAYMENT finds the POID, it adds it to the bills array in its output flist. The information in the output flist is passed to PCM_OP_PYMT_SELECT_ITEMS by PCM_OP_PYMT_COLLECT.
3. If the payment is not associated with a valid bill number, the opcode searches for a bill whose total due amount matches the payment amount. The search is restricted to bills linked to the account with which the payment is associated.

> **Note:**
>
> By default, this search is disabled.

Reversing Payments

Payment reversal batches are processed by PCM_OP_BILL_REVERSE.

> **Note:**
>
> To reverse payments, PCM_OP_BILL_REVERSE calls PCM_OP_BILL_REVERSE_PAYMENT. PCM_OP_BILL_REVERSE is the recommended opcode to use. It is a wrapper for PCM_OP_BILL_REVERSE_PAYMENT. Custom client applications should not directly call PCM_OP_BILL_REVERSE_PAYMENT.

PCM_OP_BILL_REVERSE performs the following operations:

1. Opens a transaction and checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled.
2. Validates that the PIN_FLD_FLAGS field is not present in the input flist or, if the flag is present, that it is *not* set to PIN_REVERSE_FLAG_REVERSE_AS_UNALLOCATED (**1**). If the flag is set, the operation fails because payments that were removed already from BRM as unallocatable cannot also be reversed.
3. Checks the PIN_FLD_STATUS field of each reversal associated with the payment being reversed.
4. Calls PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH, which performs the following operations:

- Validates that the payment does *not* have a SUB_TRANS_ID value, and is therefore the original payment. If the payment has a SUB_TRANS_ID value, the operation will fail.
- Finds all distributed payment events that have the same SUB_TRANS_ID value as this payment's TRANS_ID value, and have not been reversed already due to the recycling process.
- Assigns a reversal TRANS_ID value to each payment returned by the search and populates the reversal flist with each TRANS_ID value.

5. Checks the PIN_FLD_STATUS field of each **/event/billing/reversal** object associated with the payment being reversed to ensure that no part of the payment has been removed from suspense as unallocatable.
6. Calls PCM_OP_BILL_REVERSE_PAYMENT to reverse the list of payments.
   - If the payment was originally made to a customer account, the list includes any recycled payment generated if the payment was moved into the suspense account after it posted to the customer account.
   - If the payment was originally made to the suspense account, the list contains all payments generated from the original payment, including distributed payments and any payment remaining in the suspense account.

     When performing reversals during payment suspense recycling, PCM_OP_BILL_REVERSE_PAYMENT must be called by PCM_OP_PYMT_RECYCLE_PAYMENT to ensure that only payments with a SUB_TRANS_ID value of NULL can be reversed directly. The reversal of recycled payments is disallowed if the reversal is not called by PCM_OP_PYMT_RECYCLE_PAYMENT. Only suspended payments and payments in customer accounts which have not been recycled can be reversed directly by PCM_OP_BILL_REVERSE_PAYMENT.
   - If the reversal is called for a SEPA payment transaction, the opcode creates the **/sepa/dd/reversal** object only if the original payment request is in the REQUESTED status. If the charge event, **/event/billing/charge/sepa**, does not exist, it records the reversal only (the **/sepa/dd/reversal** object is not created assuming that the payment request has not been sent to the bank).

   The sum of all the payments reversed in this operation should equal the amount of the original payment.

   PCM_OP_BILL_REVERSE_PAYMENT verifies that the reversal operation was successful for all of the payments. It uses the PIN_FLD_SESSION_OBJ field in the input flist to reference the reversal batch event.

   It also checks the value of the **PaymentIncentive** business parameter to determine whether payment incentives are enabled. If so, PCM_OP_BILL_REVERSE_PAYMENT calls PCM_OP_PYMT_REVERSE_INCENTIVE, which removes the payment incentive trigger in the bill unit, thus eliminating the payment incentive.
7. Populates the payment batch with the sum of the reversal flist.
8. Returns the reversal information for all the payments in the PIN_FLD_MULTI_RESULTS array.

The PIN_FLD_RESULTS field of the output flist indicates whether the reversal was successful. Direct reversal is not allowed if either of the following conditions is true:

- The payment has a SUB_TRANS_ID value and, therefore, is *not* an original payment.

- PCM_OP_BILL_REVERSE is called from PCM_OP_PYMT_RECYCLE_PAYMENT. In this case, a reversal takes place, but it is not a direct reversal. See "How Suspended Payments Are Reversed".

To customize how written off payments are reversed, use PCM_OP_BILL_POL_REVERSE_PAYMENT.

If you have Payment Suspense Manager enabled, you can reverse only original payments. For information on payment reversals that occur during payment suspense processing, see "How Suspended Payments Are Reversed".

Refunding Payments

PCM_OP_BILL_ITEM_REFUND is used to create a refund item for a bill or bill unit. POIDs passed in to either the PIN_FLD_BILL_OBJ field or the PIN_FLD_BILLINFO_OBJ field specify the billing entity to receive the refund.

After the refund items are created, BRM uses PCM_OP_PYMT_COLLECT to refund the payment amount to the account. For BRM-initiated payments, use the **pin_collect** utility to refund payments based on the payment method. For externally initiated payments, use Billing Care or Customer Center to perform the refund operation.

Before it refunds a payment, BRM determines whether Payment Suspense Manager is enabled. If so, it checks the account POID in the input flist against the account POID in the **/config/psm** object to see whether the account is the suspense account. If the two POIDs match, the opcode generates an error.

There are two ways to run PCM_OP_BILL_ITEM_REFUND: *calculate-only* mode or *regular* mode.

In calculate-only mode, PCM_OP_BILL_ITEM_REFUND:

- Computes the total refund amount without actually creating the refund item or committing any changes to the database.
- It shows the amount that a CSR can refund, based on both credit and debit items.

In the regular mode, PCM_OP_BILL_ITEM_REFUND does the following:

- Creates a refund item for the account if it has an open credit. Otherwise the refund fails and an error message is returned.
- Transfers amounts from any open credit items to the new refund item, and closes the credit items.

> **Note:**
>
> For any transfers, PCM_OP_BILL_ITEM_REFUND calls PCM_OP_BILL_ITEM_TRANSFER.

- Transfers amounts from the bill unit or bill to any open debit items.

  Each transfer from the new refund item to debit items decreases the value of the new refund item.
- Creates the **/item/refund** object, which contains the total credit amount.
- Closes these credit and debit items.
- Returns the refundable amount in the **/item/refund** object.

If you use a custom program to perform refunds, you can put the program name in the input flist optional field PIN_FLD_PROGRAM_NAME. If this field is used to contain the name of the program refunding a customer's account, the program name is recorded in the events associated with an item refund. If the program name is not specified, the default value, Refund Opcode, is used.

> **Note:**
>
> - You cannot refund suspended payments.
> - You cannot reverse a refund. If you refund a customer's account by mistake, adjust the account for the refunded amount.

Writing Off Payments

See the following topics:

- About Initiating Write-Offs
- About Account Write-Offs
- About Bill Unit Write-Offs
- Performing Write Offs

Finding Payment Info

To find **/payinfo** objects that belong to an account, use PCM_OP_CUST_FIND_PAYINFO.

This opcode is given the account POID and returns the information from the storable **/payinfo** object.

Adding a Custom Payment Method

To add a payment method:

- Update the **/config/payment** object.
- Modify the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to validate the custom payment method. For example, add code for your custom payment method everywhere the opcode checks the various payment methods.

To add a custom payment method:

1. Using a text editor, edit the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode source code to add an element to the PIN_FLD_PAY_TYPES array for each new payment method.
   - For a predefined payment method, use the appropriate value as the index for that element of the PIN_FLD_PAY_TYPES array.
   - For a custom payment method, define a payment method in a header file and use that value as the index of the new element in the PIN_FLD_PAY_TYPES array.

> **Note:**
>
> To avoid conflicts with payment IDs used by BRM, define custom payment methods with an element ID of 10100 or higher.

2. Save and close the file.
3. Use the **testnap** utility to load the **/config/payment** object into the database. See "Testing Your Applications and Custom Modules" in *BRM Developer's Guide*.

Creating a /config/payment Object

Instead of changing the default **/config/payment** object, you can configure your own **/config/payment** object and load it into the database using the **testnap** utility. See *BRM Developer's Guide*.

After you create the **/config/payment** object, update the **config_payment** entry in the CM **pin.conf** file with the POID of the new object:

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

   ```
   - fm_pymt config_payment database_number /config/payment 200
   ```

   where database_number is the database number of the BRM database. By default, this number is **0.0.0.1**. Replace **200**, the default POID of the **/config/payment** object, with the POID of the new object.
3. Save and close the file.
4. Stop and restart the CM.

Viewing Payment Information for a Custom Payment Method

To view payment information for custom payment methods, run the **pin_collect** utility for that payment method. The **pin_bill_day** utility calls **pin_collect** for credit card payment methods and **pin_inv_accts** for invoice payment methods. You can make a call to **pin_collect** in **pin_bill_day** for each new custom payment method.

For more information, see *BRM Configuring and Running Billing* guide and *BRM Configuring and Collecting Payments* guide.

Configuring Payment Center for Custom Payment Methods

If you create custom payment methods for your BRM system, you must customize Payment Center to handle them. This overview procedure describes how to create custom storable classes and fields and enable Payment Center to handle them.

> **Note:**
>
> Before customizing your payment functionality, you should be familiar with the Java PCM and the BRM Storable Class Editor, which you use to create custom storable classes and fields. For background information on creating custom storable classes and fields, see "Creating Client Applications by Using Java PCM" and "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

1. Complete the following tasks by using Storable Class Editor:
   a. Create your storable classes and fields in the Java PCM package.
   b. Create source files for your custom fields.

   > **Note:**
   >
   > Storable Class Editor creates a C header file called **cust_flds.h**, a Java properties file called **InfranetPropertiesAdditions.properties**, and a Java source file for each custom field.

2. In the directory in which Storable Class Editor created the Java source files, compile the source files:

   ```
   javac -d . *.java
   ```

3. Package the new storable class files into a JAR file. For example:

   ```
   jar cvf customfields.jar *.class
   ```

4. Copy the contents of the **InfranetPropertiesAdditions.properties** file and paste it into the Payment Center **Infranet.properties** file. By default, this file is located in the **C:\Program Files\Portal Software\PaymentCtr\PaymentCenter** directory.
5. Append the location of the JAR file to the PAYCTRCP environment variable path. For example:

   ```
   ;.;C:\Program Files\Portal Software\PaymentCtr\customfields.jar;
   ```

How Opcodes Read the Payment Method

Payment methods are used in the following objects:

- The PIN_FLD_PAY_TYPE field in the **/billinfo** object defines the payment method for an account's bill. All defined payment methods are stored in the **/config/payment** object.
- When you add a new payment method to your BRM system, you must update the PIN_FLD_PAY_TYPES array in the **/config/payment** object. The PIN_FLD_PAY_TYPES array index corresponds to an entry in the **pin_pymt.h** file, where payment methods are defined.

When you create a payment method, you must consider the opcodes that might use that payment method. For example, if you change a customer's payment method to a custom payment type, you must add the new payment method, when first used, to the input flist of the opcodes that use the PIN_FLD_PAY_TYPE field. This is required if an invoice customer makes a one-time payment by using a new credit card. If the new payment method is SEPA, you must

add the payment method to the account before it can be used. This is not required if the payment method was defined when the account was created.

The following lists the opcodes that use the PIN_FLD_PAY_TYPE field:

- PCM_OP_PYMT_CHARGE
- PCM_OP_PYMT_CHARGE_CC
- PCM_OP_PYMT_COLLECT
- PCM_OP_BILL_GROUP_DELETE_MEMBER
- PCM_OP_BILL_MAKE_BILL
- PCM_OP_BILL_MAKE_BILL_NEW
- PCM_OP_BILL_MAKE_BILL_ON_DEMAND
- PCM_OP_BILL_MAKE_BILL_RCV_PAYMENT
- PCM_OP_PYMT_RECOVER
- PCM_OP_BILL_REVERSE
- PCM_OP_BILL_REVERSE_PAYMENT
- PCM_OP_PYMT_VALIDATE

When you define a custom payment method, you must provide information about the payment method. Table 21-9 lists the fields in PIN_FLD_PAY_TYPES array.

Table 21-9 Fields in PIN_FLD_PAY_TYPE

Field name	Description
PIN_FLD_PAYINFO_TYPE	A string that indicates the type of **/payinfo** object to create.
PIN_FLD_PAYMENT_EVENT_TYPE	A string that indicates the type of **/event** object to create when payment is received.
PIN_FLD_REFUND_EVENT_TYPE	A string that indicates the type of **/event** object to create when a refund is paid.
PIN_FLD_OPCODES	An array that contains the following information about an opcode to be associated with this pay type: • PIN_FLD_NAME The name of the opcode to be associated with this pay type. • PIN_FLD_OPCODE The number of the opcode to be associated with this pay type. • PIN_FLD_EVENT_TYPE A string that indicates the type of **/event** object to create. • PIN_FLD_FLAGS Flags passed to opcodes when run.

Table 21-10 lists the PIN_FLD_OPCODES array indexes-to-opcode mapping.

Table 21-10 Index to Code Mapping for PIN_FLD_OPCODES

Value/Element ID	Opcode type
0	PIN_PAY_VALIDATE
1	PIN_PAY_CHARGE
2	PIN_PAY_RECOVER
3	PIN_BILL_REVERSAL

Processing Payment Fees

When processing payment fees:

- Payment fees are applied by PCM_OP_PYMT_APPLY_FEE. See "Applying Payment Fees".
- To customize payment fees, use PCM_OP_PYMT_POL_APPLY_FEE. See "Customizing Payment Fees".
- To store additional information for payment fees, see "Storing Additional Information with Payment Fees" for information.

Applying Payment Fees

Payment fees are applied by PCM_OP_PYMT_APPLY_FEE. This opcode creates payment fees for payments that fail, for example, due to insufficient account funds or an expired credit card. It calls PCM_OP_ACT_USAGE to create the payment fee event to be rated.

When a failed payment is posted in BRM, it is recorded in BRM with a balance impact of **0**, and is identified by a transaction ID, a failure status, and a reason ID for the failure. BRM uses the failure status and reason ID to determine whether to apply payment fees when the payment is posted. Payment fees can be applied only to payments that have a PIN_FLD_STATUS value of **PIN_PYMT_FAILED** in the **/event/billing/payment/failed** object.

PCM_OP_PYMT_APPLY_FEE is called by PCM_OP_PYMT_COLLECT.

The behavior of PCM_OP_PYMT_APPLY_FEE is determined by the PIN_FLD_STATUS field passed in on the input flist. Payments are eligible to receive payment fees if they have a PIN_FLD_STATUS value >= **PIN_PYMT_FAILED** and < **PIN_PYMT_STATUS_MAX**. The numeric range for financially failed payments is 30-44.

The normal flow of PCM_OP_PYMT_APPLY_FEE is as follows:

1. Checks the input flist for the status of the payment and the reason ID that describes why the payment failed.
2. Calls PCM_OP_PYMT_POL_APPLY_FEE to perform custom checks before the failed payment fee is applied. See "Customizing Payment Fees".

   When it returns the output flist, PCM_OP_PYMT_APPLY_FEE validates the information, and creates the failed payment fee events for all failed payments based on the information. The default value in the PIN_FLD_BOOLEAN field on the output flist is **0**, which specifies that the fee is created.
3. If a payment fails, records the **/event/billing/payment/***payment_type* event.

> **Note:**
>
> If the payment is an unconfirmed payment, records the **/event/billing/payment/failed** event.

4. Creates the **/event/billing/fee/failed_payment** event.

PCM_OP_PYMT_APPLY_FEE provides feedback on its success or failure through the PIN_FLD_RESULTS array in the output flist. The value in the PIN_FLD_RESULTS field specifies whether the payment fee event was created. A value of **0** signifies that the payment fee event was created and the payment fee applied. A nonzero value signifies that the payment fee event was not created.

In the case of a write-off reversal, the output flist sends a results array of reversal events and tax events (if created) that were passed in by PCM_OP_AR_REVERSE_WRITEOFF.

> **Note:**
>
> Failed payments can only be applied to account numbers that already exist in the BRM database.

The PIN_FLD_EVENTS array of the output flist stores the POID of the failed payment fee event or write off event, if one is created. The PIN_FLD_EVENTS array is contained in the PIN_FLD_RESULTS array.

Flags are not used directly by PCM_OP_PYMT_APPLY_FEE. They are passed in from PCM_OP_PYMT_COLLECT for PCM_OP_PYMT_SELECT_ITEMS.

Customizing Payment Fees

You can define additional rules for payment fee processing by configuring PCM_OP_PYMT_POL_APPLY_FEE. This opcode is called by PCM_OP_PYMT_APPLY_FEE.

This opcode also enhances the **/event/billing/fee/failed_payment** object by providing additional fields that are recorded in the object.

PCM_OP_PYMT_POL_APPLY_FEE enables you to customize payment fees by preprocessing, filtering, and extending the information available in failed payment fee events. For example:

- You can charge different fee amounts based on thresholds, or on the reason ID associated with a failed payment.
- You can use payment attributes such as the customer segment, payment method, payment channel, or a combination of these attributes to charge payment fees. You create the filters by passing the values in the PIN_FLD_EXTENDED_INFO substruct or the PIN_FLD_CHARGES array.

The default behavior of PCM_OP_PYMT_POL_APPLY_FEE is determined by the PIN_FLD_STATUS and PIN_FLD_REASON_ID fields passed in on the input flist. It enhances the input flist by adding fields to filter and extend the information available in the payment fee events.

> **Note:**
>
> If the transaction ID, status or reason ID is missing from the actual payment, it can be retrieved from the payment batch header.

Using Custom Reason Codes to Customize Payment Fees

You can use custom reason codes to apply reasons to payment failures, and then customize payment fees based on those reasons.

To use custom reason codes, you must customize the payment gateway to and your third-party payment application to identify failed payments and send reason codes with the payment information. When a failed payment is received, the reason code is mapped to the reason code ID defined in the BRM database.

You define reason codes in the **reasons**.*locale* file and load them into the BRM database as a */* **strings** object. The file contains instructions on how to add the new domain *Reason Codes - Payment Failure Reasons*. For example:

```
DOMAIN = "Reason codes-Payment Failure Reasons";
STR
     ID = 1001 ;
     VERSION = 13 ;
     STRING = "Invalid Credit Card";
END
```

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

To define reason codes for failed payments, you edit the **reasons.en_US** sample file in the *BRM_home***/sys/msgs/reasoncodes** directory. You then use the **load_localized_strings** utility to load the contents of the file into the **/strings** objects.

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

```
load_localized_strings reasons.locale
```

> **Note:**
>
> - If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see Locale names.
> - If a failed payment is loaded into BRM with an invalid reason code, payment fees are not applied.

For information on loading the **reasons.**locale file, see *BRM Developer's Guide*. For information on creating new strings for this file, see *BRM Developer's Guide*.

Customization example: Charging a fee based on the customer segment

The following opcode customization is used to control which failed payment fee is assigned to an account based on the account's customer segment:

```
If (customer_segment = "early bill payer")
Then set PIN_FLD_BOOLEAN to False
END
```

The PIN_FLD_BOOLEAN value of **False** specifies that the fee event is not created. When payments are posted, BRM uses the customer segment ID to determine if a payment fee is charged.

You can also use the PIN_RESULT_PASS and PIN_RESULT_FAIL return values in PCM_OP_PYMT_POL_APPLY_FEE to configure whether payment fees are applied.

Storing Additional Information with Payment Fees

You can store additional information for payment fees by extending the **/event/billing/fee/failed_payment** storable class. Use the PIN_FLD_FAILED_PAYMENT_FEE field inside the PIN_FLD_EXTENDED_INFO substruct. This enables you to record additional criteria you defined to create the payment fee.

You can then use the PIN_FLD_REASON_ID field in the input flist to configure fees based on this value of the PIN_FLD_FAILED_PAYMENT_FEE field. It contains the reason for failure that was sent by the payment processor for failed credit card and direct debit transactions.

You can then charge payment fees based on custom payment attributes such as currency type:

1. Extend the **/event/billing/fee/failed_payment** storable class with the new attributes by using Developer Center.
2. Customize PCM_OP_PYMT_POL_APPLY_FEES to pass the value in the PIN_FLD_CHARGES array or the PIN_FLD_FAILED_PAYMENT_FEE field in the PIN_FLD_EXTENDED_INFO substruct.
3. Use PDC or Pricing Center to create the charges for the new attribute values.

Processing Payment Incentives

Payment incentives are implemented by calling PCM_OP_PYMT_PROVISION_INCENTIVE and PCM_OP_PYMT_GRANT_INCENTIVE.

PCM_OP_PYMT_PROVISION_INCENTIVE evaluates a payment to determine whether a payment incentive should be provisioned and, if so, sets the payment incentive trigger. If a payment incentive is triggered, PCM_OP_PYMT_GRANT_INCENTIVE performs the incentive.

PCM_OP_PYMT_PROVISION_INCENTIVE is called by PCM_OP_BILL_ITEM_TRANSFER immediately after payment allocation, provided BRM is configured for payment incentives. PCM_OP_PYMT_PROVISION_INCENTIVE determines whether the payment resulted in an early, in-full settlement of the last bill. If so, the current bill may be eligible for a payment incentive. This opcode creates a trigger for payment incentive processing to apply an incentive.

For more information, see "Triggering Payment Incentives" and "Granting Payment Incentives".

Triggering Payment Incentives

PCM_OP_PYMT_POL_PROVISION_INCENTIVE is called by PCM_OP_BILL_ITEM_TRANSFER immediately after payment allocation, provided BRM is configured for payment incentives.

You can customize PCM_OP_PYMT_POL_PROVISION_INCENTIVE to provide the timestamp from a field other than PIN_FLD_END_T (for example, PIN_FLD_EFFECTIVE_T) or to apply business logic that determines the payment date. For example, you can customize PCM_OP_PYMT_POL_PROVISION_INCENTIVE to use the payment receipt date as the payment timestamp for all credit card payments and three days after the payment receipt date for all check payments.

PCM_OP_PYMT_PROVISION_INCENTIVE determines whether the payment resulted in an early, in-full settlement of the last bill. If so, the current bill may be eligible for a payment incentive and PCM_OP_PYMT_POL_PROVISION_INCENTIVE creates a trigger for payment incentive processing to apply an incentive.

PCM_OP_PYMT_POL_PROVISION_INCENTIVE is called by PCM_OP_PYMT_PROVISION_INCENTIVE to provide the payment date that is used when determining whether the bill was paid on time. It receives the POID of the **/event/billing/payment** object in the input flist and reads this object to determine the payment date. By default, PCM_OP_PYMT_POL_PROVISION_INCENTIVE reads the PIN_FLD_END_T field to obtain the timestamp. If it finds a payment date it's configured to provide, it returns the date to PCM_OP_PYMT_PROVISION_INCENTIVE, which performs the following tasks:

- Retrieves the bill for which the payment allocation was made.
- Determines if the bill was the last bill.
- Compares the timestamp provided by the opcode to the due date for the payment.

PCM_OP_PYMT_PROVISION_INCENTIVE performs the following functions:

1. It calls PCM_OP_PYMT_POL_PROVISION_INCENTIVE to determine if the PIN_EFFECTIVE_T field or any other customizable field contains a payment timestamp. See "Triggering Payment Incentives".
2. It retrieves all of the bills and determines whether each of the bills is from the last billing cycle or a prior cycle. PCM_OP_PYMT_PROVISION_INCENTIVE is only concerned with the bills from the last billing cycles; none of the other bills qualify for early incentives.
3. It reads the Due and Due Time in each **/bill** object. PCM_OP_PYMT_PROVISION_INCENTIVE uses this information along with the timestamp in the PIN_FLD_END_T field from the **/event/billing/payment** object or a timestamp from the policy opcode to determine whether the payment for a given bill was allocated in full and early. PCM_OP_PYMT_PROVISION_INCENTIVE must find the following conditions:
   - PIN_FLD_DUE must be 0, indicating that there are no more payments due for the bill, and it was paid in full.
   - The timestamp in PIN_FLD_END_T or the timestamp provided by the policy opcode must be earlier than or the same as the one in PIN_FLD_DUE_T, indicating that the payment was allocated before or at the same time as the due time. BRM considers both of these conditions to be indications of an early payment.

   BRM always attempts to use whatever timestamp the policy opcode provides. PCM_OP_PYMT_PROVISION_INCENTIVE only uses PIN_FLD_END_T if the policy

opcode does not return a timestamp or returns PIN_FLD_END_T instead of some other timestamp.

4. If the payment meets these conditions, PCM_OP_PYMT_PROVISION_INCENTIVE modifies the **/billinfo** object by setting its PIN_FLD_PAYMENT_EVENT_OBJ field to the POID of the payment event that resulted in early, in-full payment. This acts as a trigger for granting the payment incentive during the billing run.
5. It returns a list of bill units to which it added the payment incentive.

Granting Payment Incentives

PCM_OP_PYMT_GRANT_INCENTIVE is called by PCM_OP_BILL_MAKE_BILL as part of the billing run. PCM_OP_PYMT_GRANT_INCENTIVE grants payment incentives based on:

- Whether the account has purchased a payment incentive subscription charge offer or the account is eligible for a system charge offer that includes a payment incentive.
- Whether the payment incentive trigger is set in the bill unit.
- Conditions specified in the charge.
- Any additional conditions specified in PCM_OP_PYMT_POL_GRANT_INCENTIVE.

PCM_OP_PYMT_GRANT_INCENTIVE performs the following functions:

1. If the bill qualifies for a payment incentive, the opcode uses information from the **/account** and **/event/billing/payment** objects to enrich the input flist with the payment method, payment channel, and customer segment list. It also includes:
   - The total for the current bill calculated during the current billing run.
   - The total for the last bill, as determined from the **/bill** object for that bill.

   By default, both these totals are after-tax amounts.
2. It calls PCM_OP_PYMT_POL_GRANT_INCENTIVE to determine whether it must enrich the input flist with any extra fields, and validates the fields returned by the policy opcode.
3. It creates an **/item/incentive** object and an **/event/billing/incentive** object for the bill. In addition to the payment method, payment channel, and so on, the **/event/billing/incentive** object contains any fields specified in PCM_OP_PYMT_POL_GRANT_INCENTIVE, provided the object has been suitably extended.
4. It sends the event to the rating opcodes to calculate the payment incentive for each bill. BRM applies the balance impact of the payment incentive event to the default balance group of the bill unit.
5. It clears the payment incentive trigger in the PIN_FLD_PAYMENT_EVENT_OBJ field of the **/billinfo** objects of each affected bill, returning this field to a null value.

Two other opcodes are used for payment incentives:

- PCM_OP_PYMT_PROVISION_INCENTIVE triggers payment incentives. See "Triggering Payment Incentives".
- PCM_OP_PYMT_REVERSE_INCENTIVE reverses payment incentives. See "Reversing Payment Incentives".

Reversing Payment Incentives

PCM_OP_PYMT_REVERSE_INCENTIVE reverses a payment incentive if it has not yet been granted.

PCM_OP_PYMT_REVERSE_INCENTIVE is called by PCM_OP_BILL_REVERSE_PAYMENT as part of payment reversal, provided BRM is configured for payment incentives. It determines whether payment is being reversed for a bill that had a payment incentive either provisioned or granted. If so, it either deactivates the payment incentive trigger or, for payment incentives that have already been granted, issues warnings and flags the reversal event so that you can initiate manual processing.

PCM_OP_PYMT_REVERSE_INCENTIVE performs the following functions:

1. It retrieves all bills affected by the reversal and searches for the associated **/billinfo** objects.
2. For each **/billinfo** object, it checks the PIN_FLD_PAYMENT_EVENT_OBJ field to determine whether a payment incentive has been provisioned but not yet granted. If so, it resets this field, removing the POID of the payment event, which automatically reverses the payment incentive.
3. If the PIN_FLD_PAYMENT_EVENT_OBJ field does not indicate that a payment incentive has been provisioned, the opcode searches all **/event/billing/incentive** objects for the account to determine whether any of them are associated with the bill for which payment is being reversed.

   The existence of this object for a given bill means that BRM already granted a payment incentive for the bill. In this case, the opcode generates a warning in the **cm.pinlog** file to indicate that manual adjustment is required.

Customizing Payment Incentives

You can configure BRM to grant payment incentives. See *BRM Configuring and Collecting Payments*.

To customize payment incentives, read the following:

- To customize how payment incentives are triggered (for example, by date), use the PCM_OP_PYMT_POL_PROVISION_INCENTIVE policy opcode. See "Customizing How to Trigger Payment Incentives".
- To customize how to grant payment incentives, use the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode. See "Customizing How to Grant Payment Incentives".
- To manually reverse payment incentives, see "Manually Reversing a Payment Incentive".

Customizing How to Trigger Payment Incentives

You can configure the PCM_OP_PYMT_POL_PROVISION_INCENTIVE policy opcode to determine the payment date that should be considered when provisioning incentives.

For example, you can customize this opcode to use the payment receipt date as the payment timestamp for all credit card payments, and 3 days after the payment receipt date for all check payments.

You can also create custom payment objects that use fields other than a payment due date. In this case, you would customize this policy opcode to read these fields and provide them as output for PCM_OP_PYMT_PROVISION_INCENTIVE.

Customizing How to Grant Payment Incentives

By default, you can set up a charge so that BRM considers three attributes when determining whether to apply a payment incentive:

- Customer segment
- Payment channel
- Payment method

You can broaden this scope by customizing the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode. When you customize this opcode, you must also extend the **/event/billing/incentive** storable class so that the object records all the criteria considered for the payment incentive. The additional fields are then included in the choices you can make when creating charge selectors in PDC or Pricing Center. Therefore, you can use these additional fields as criteria when defining attribute combinations that result in payment incentives.

The contents of the **/event/billing/incentive** object determine:

- The list of attributes that can be considered when setting up a charge. The pricing expert associates combinations of these attributes with specific charges when creating the payment incentive charge offer.
- The information BRM compares with the attribute combinations defined for the payment incentive charge offer. If BRM finds a match, it rates the payment incentive event according to the charge for the matching combination.

For example, to award a payment incentive to premium customers who pay in a certain currency, you perform two tasks:

- Customize PCM_OP_PYMT_POL_GRANT_INCENTIVE by adding the PIN_FLD_SERVICE_OBJ and PIN_FLD_CURRENCY fields to the input flist. The opcode then includes the service type for the account and the currency type in the output flist. As a result, BRM is able to consider the service type and currency along with the usual criteria when determining whether to apply the incentive and when calculating the payment incentive.
- Extend the **/event/billing/incentive** storable class by adding the PIN_FLD_SERVICE_OBJ and PIN_FLD_CURRENCY fields. BRM then includes the service type and currency in the event object each time it's created. When you create columns in the charge selector, you can select these two fields along with the default fields.

In addition to performing this type of customization, you can also customize the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode to do the following:

- Provide special types of incentives such as free gifts. In this case, you modify the opcode so that it calls the opcodes that process and control charge offer purchases, for example, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. Then, you create a bundle that includes an item charge offer that awards the free gift.
- Develop an application to count the number of subscription services for an account and customize PCM_OP_PYMT_POL_GRANT_INCENTIVE to include this information in the enriched flist.
- Customize which customer segment to use. You can also customize this opcode to select a different customer segment from PIN_FLD_CUSTOMER_SEGMENT_LIST. By default, PCM_OP_PYMT_POL_GRANT_INCENTIVE selects the first customer segment from the PIN_FLD_CUSTOMER_SEGMENT_LIST it gets from PCM_OP_PYMT_GRANT_INCENTIVE. The customer segment returned by PCM_OP_PYMT_POL_GRANT_INCENTIVE is the one that BRM uses as a filtering attribute during payment incentive calculation.

Manually Reversing a Payment Incentive

When a payment is reversed, BRM reverses any payment incentive provisioning triggers created at payment time, provided the payment was for the last bill. If payment was reversed

for an earlier bill, BRM has already applied the incentive and does not reverse it. In this case, you must perform a manual account adjustment through your CRM client application. This type of adjustment debits account balances rather than crediting them.

BRM identifies payment incentives that need manual reversal in the **cm.pinlog** file, which lists any cases where a payment incentive reversal failed. Searching the **cm.pinlog** for this information can be time consuming. Therefore, you should consider customizing this process in one of the following ways:

- Write your own reporting application that creates a list of all bills that meet the following two conditions:
  - The bill had a payment incentive that was not only provisioned, but also granted.
  - The payment reversal was for the bill before the one that had the payment incentive granted.

  Operations personnel can use this report to identify the adjustments they must perform.
- Create your own custom opcode or customize the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode so that it checks the reporting conditions discussed above and alerts the CRM client application whenever both conditions are true. Then, enable event notification and add the following information to your system's event notification list so that BRM calls PCM_OP_ACT_POL_EVENT_NOTIFY (opcode number 301) each time an **/event/billing/reversal/*** event is generated during payment reversal:

  ```
  # comment, if any
  301    0    /event/billing/reversal/cc
  301    0    /event/billing/reversal/check
  301    0    /event/billing/reversal/dd
  301    0    /event/billing/reversal/payorder
  301    0    /event/billing/reversal/postalorder
  301    0    /event/billing/reversal/transfer
  ```

  For more information, see "Using Event Notification" in *BRM Developer's Guide*.

  In addition, you must customize your middleware and CRM applications to process the notification correctly and issue appropriate messages to operations personnel.

Managing Top-Ups

See the following for information about managing top-ups in your client applications:

- Setting Up Top-Up Information in an Account
- Processing Top-Ups
- Implementing Top-Ups in Custom Client Applications
- Setting an Account's Sponsored Top-Up Member Status and PIN
- Offering Discount Incentives with Top-Ups

Setting Up Top-Up Information in an Account

To create or modify an account's top-up information, PCM_OP_CUST_SET_TOPUP calls one of the following opcodes:

- PCM_OP_CUST_CREATE_TOPUP
- PCM_OP_CUST_MODIFY_TOPUP

If successful, the PCM_OP_CUST_SET_TOPUP output flist contains the PIN_FLD_POID set to the POID of the created or modified **/topup** object.

If unsuccessful, the PCM_OP_CUST_SET_TOPUP output flist contains the following:

- PIN_FLD_FIELD_NUM set to the field that failed.
- PIN_FLD_TYPE set to the type of field that failed.
- PIN_FLD_RESULT set to the validation error code.

For standard top-ups, an account's top-up information is stored in one object (**/topup**), but for sponsored top-ups, an account's top-up information is stored in *two* objects (**/topup** and **/group/topup**). Sometimes, one of the objects must be created and the other modified. For example, to add a member account to a sponsored top-up group, BRM might need to create a **/topup** object for the member and associate it with an existing **/group/topup** object. To determine which opcode to call in such cases, PCM_OP_CUST_SET_TOPUP uses these rules:

- If at least one of the objects is being created, PCM_OP_CUST_SET_TOPUP calls PCM_OP_CUST_CREATE_TOPUP. This is true even if the other object is being modified.
- If neither object is being created, PCM_OP_CUST_SET_TOPUP calls PCM_OP_CUST_MODIFY_TOPUP.

Preparing an Account's Top-Up Information

To prepare top-up information for an account, PCM_OP_CUST_CREATE_TOPUP and PCM_OP_CUST_MODIFY_TOPUP call PCM_OP_CUST_POL_PREP_TOPUP. For example, you can customize PCM_OP_CUST_POL_PREP_TOPUP to enable member accounts to change their top-up PINs and membership status. The policy opcode prepares information required to perform one of these tasks:

- **Create a standalone /topup object for standard top-ups.** This occurs when the following information is *not* passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A **/topup** object POID
  - A sponsored top-up group owner account POID
- **Modify a standalone /topup object for standard top-ups.** This occurs when the following information is passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A **/topup** object POID

  But this information is *not* passed to it:
  - A sponsored top-up group owner account POID
  - A **/group/topup** POID
- **Create one or both objects (/topup and /group/topup) for sponsored top-ups.** This occurs when the following information is passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A sponsored top-up group owner account POID

  But this information is *not* passed to it:
  - A **/topup** object POID
  - A **/group/topup** POID

> **Note:**
>
> Before creating a **/group/topup** object, the opcode checks for an existing **/group/topup** object that matches the criteria in its input flist. For more information, see "Finding Sponsored Top-Up Groups".

- **Modify both objects (/topup and /group/topup) for sponsored top-ups.** This occurs when the following information is passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A **/topup** object POID
  - A **/group/topup** POID

Additional Preparation for Sponsored Top-Ups

For sponsored top-ups, PCM_OP_CUST_POL_PREP_TOPUP also prepares this information:

- **If the group owner account did** not **initiate the object creation or modification,** the policy opcode sets the following values in its output flist:
  - PIN_FLD_STATUS (member account's group membership status) = the value associated with the **PIN_STATUS_INACTIVE** status in the *BRM_home***/include/ops/pcm.h** header file
  - PIN_FLD_PIN (member account's top-up PIN) = **NULL**
- **If the group owner account** did **initiate the object creation or modification,** the policy opcode does the following:
  - If the status of the member account's group membership is not specified in the input flist, sets it to the value associated with the **PIN_STATUS_ACTIVE** in the *BRM_home***/include/ops/pcm.h** header file
  - **(Creation only)** If the group name is not specified in the input flist, sets the name to **default**

Validating an Account's Top-Up Information

To validate the top-up information prepared for an account, PCM_OP_CUST_CREATE_TOPUP and PCM_OP_CUST_MODIFY_TOPUP call PCM_OP_CUST_POL_VALID_TOPUP. See "Preparing an Account's Top-Up Information" for more information on preparing top-up data. The policy opcode performs these tasks:

- Verifies that the status of the account to be debited for each top-up (the paying account) is *active*.
- Verifies that the standard or sponsored top-up amount is less than or equal to the corresponding top-up cap.
- **(Sponsored top-ups only)** Verifies that the member is not trying to join a group that it owns.
- **(Sponsored top-ups only)** Verifies that the prospective member's *account* is not closed.
- **(Sponsored top-ups only)** Verifies that the prospective member is not a member of any other sponsored top-up group.

You can customize PCM_OP_CUST_POL_VALID_TOPUP to change the way it validates the PCM_OP_CUST_POL_PREP_TOPUP output flist.

In its own output flist, PCM_OP_CUST_POL_VALID_TOPUP returns a PIN_FLD_RESULT value that is associated with one of the following values:

- **PIN_RESULT_PASS** (validation succeeded)
- **PIN_RESULT_FAIL** (validation failed)

PCM_OP_CUST_POL_VALID_TOPUP is not called by any opcode.

Finding Top-Up Events

PCM_OP_PYMT_FIND_TOPUP_EVENTS finds the **/event/billing/adjustment/account** event associated with sponsored top-ups.

By default, this opcode returns data from all the fields in an event. To return data from only particular event fields, specify the fields in the PIN_FLD_RESULTS array in this opcode's input flist.

Creating an Account's Top-Up Information

If validation succeeds, the validated information is used to create top-up information. See "Validating an Account's Top-Up Information" for more information about validation.

PCM_OP_CUST_CREATE_TOPUP uses the validated information to perform one of these operations:

- If the information does not include the POID of a sponsored top-up group owner account, the opcode:
  - Creates a standalone **/topup** object that contains information about the type of top-up to perform.
  - For recurring top-ups, calls PCM_OP_PYMT_TOPUP to perform the first top-up. See "How BRM Handles Recurring Standard Top-Ups".
  - Generates an **/event/notification/topup/create** event.
- If the information includes the POID of a sponsored top-up group owner account and **/group/topup** and **/topup** POID types, the opcode creates a sponsored top-up relationship as follows:
  1. Creates a **/group/topup** object for the owner account
  2. Creates a **/topup** object for the member account
  3. Associates the new **/topup** object with the new **/group/topup** object
- If the information includes the POID of a sponsored top-up group owner account, the POID of an existing **/group/topup** object, and a **/topup** POID type, PCM_OP_CUST_CREATE_TOPUP creates a sponsored top-up relationship as follows:
  1. Creates a **/topup** object for the member account
  2. Associates the new **/topup** object with the existing **/group/topup** object
- If the information includes the POID of a sponsored top-up group owner account, a **/group/topup** POID type, and an existing **/topup** object, PCM_OP_CUST_CREATE_TOPUP creates a sponsored top-up as follows:
  1. Creates a **/group/topup** object for the owner account
  2. Associates the existing **/topup** object with the new **/group/topup** object

If successful, the PCM_OP_CUST_CREATE_TOPUP output flist contains the following:

- PIN_FLD_POID set to the POID of the **/topup** object created

If unsuccessful, the output flist contains the following:

- PIN_FLD_FIELD_NUM set to the field that failed
- PIN_FLD_TYPE set to the type of field that failed
- PIN_FLD_RESULT set to the validation error code

Modifying an Account's Top-Up Information

If validation succeeds, the validated information is used to modify top-up information. See "Validating an Account's Top-Up Information" for more information about validation.

PCM_OP_CUST_MODIFY_TOPUP uses the validated information to perform one of these operations:

- If the information does *not* include the POID of an existing **/group/topup** object, the opcode:
  - Modifies the standalone **/topup** object.
  - For recurring top-ups, calls PCM_OP_PYMT_TOPUP to perform the first top-up. See "How BRM Handles Recurring Standard Top-Ups".
  - Generates an **/event/notification/topup/modify** event.
- If the information *includes* the POID of an existing **/group/topup** object, the opcode modifies the sponsored top-up information in the **/group/topup** and **/topup** objects.

If successful, the PCM_OP_CUST_MODIFY_TOPUP output flist contains the following:

- PIN_FLD_POID set to the POID of the **/topup** object modified

If unsuccessful, the PCM_OP_CUST_MODIFY_TOPUP output flist contains the following:

- PIN_FLD_FIELD_NUM set to the field that failed
- PIN_FLD_TYPE set to the type of field that failed
- PIN_FLD_RESULT set to the validation error code

Deleting Top-Ups

Use PCM_OP_CUST_DELETE_TOPUP to delete **/topup** objects.

PCM_OP_CUST_DELETE_TOPUP is called by PCM_OP_CUST_DELETE_ACCT.

This opcode should not be used to cancel an account's membership in a sponsored top-up group.

Finding Sponsored Top-Up Groups

When setting up sponsored top-ups, PCM_OP_CUST_POL_PREP_TOPUP uses the following information from the PCM_OP_CUST_SET_TOPUP input flist to determine whether the prospective member account can be added to an existing group:

- The group owner account POID (PIN_FLD_PARENT)
- The name of the group (PIN_FLD_NAME)

> **Note:**
>
> Each group owned by the *same* account must have a unique name. Groups owned by *different* accounts can have the same name.

If a group name is not provided, the policy opcode searches for a group by owner account POID and the ID of the balance or balances that you want to top-up in the member account (PIN_FLD_RESOURCE_ID in the LIMITS array). The search has the following results:

- If the policy opcode finds the group by name but a balance is specified in the input flist that the group does not support, the following occurs:
  - If the group owner account initiated the transaction, the balance is added to the group.
  - If the member account initiated the transaction, an error is returned.
- If the policy opcode finds the group by balance and the search returns multiple groups, the groups are listed alphabetically by PIN_FLD_NAME value and the member is added to the group at the top of the list.
- If the policy opcode fails to find a group by name or by balance, the following occurs:
  - If the group owner account has a group named **default**, the member is added to that group.
  - If the group owner account does not have a group named **default**, such a group is created based on the information in the input flist. (Each group owner can have only one sponsored top-up group named **default**.)

> **Note:**
>
> To change the way the search is performed, customize PCM_OP_CUST_POL_PREP_TOPUP.

Setting a Customer's Automatic Top-Up Threshold

You can set a fixed or percentage-based threshold below which an automatic top-up will be triggered for a customer. The thresholds are stored in **/config/credit_profile** objects, which are tied to a customer's balance group.

To set an automatic top-up threshold, use PCM_OP_BILL_SET_LIMIT_AND_CR. Specify the PIN_FLD_RESRC_FIXED_THRESHOLD or PIN_FLD_RESRC_PERC_THRESHOLD fields in the PIN_FLD_LIMIT array of the input flist, using the appropriate resource ID for the limit array.

For example, to set a threshold of $5 in USD (resource ID 840):

```
0 PIN_FLD_PROGRAM_NAME         STR [0] "calling_program"
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 371077 10
0 PIN_FLD_LIMIT              ARRAY [840] allocated 1, used 1
1       PIN_FLD_RESRC_FIXED_THRESHOLD DECIMAL [0] 5.0
```

Processing Top-Ups

BRM uses the PCM_OP_PYMT_TOPUP opcode to process all top-up payments. See the following topics:

- Triggering Top-ups
- How BRM Handles Manual Standard Top-Ups
- How BRM Handles Automatic Standard Top-Ups
- How BRM Handles Recurring Standard Top-Ups
- How BRM Handles Manual Sponsored Top-Ups
- How BRM Handles Automatic Sponsored Top-Ups

Triggering Top-ups

PCM_OP_PYMT_TOPUP is triggered as follows:

- **Manual top-ups**: When a customer or CSR uses a client application to top up a balance in an account (a manual top-up), the opcode is called by the client application.
- **Automatic standard top-ups**: When a balance in an account configured for automatic standard top-ups falls below a specified threshold, ECE calls PCM_OP_PYMT_TOPUP.
- **Recurring standard top-ups**: When you run the **pin_balance_transfer** utility with the **-standard** parameter. The utility calls the opcode to top up all accounts with a recurring top-up and a top-up due date of today.
- **Automatic sponsored top-ups**: When you run the **pin_balance_transfer** utility with the **-start** *mm/dd/yy* and **-end** *mm/dd/yy* parameters. The utility calls the opcode to top up all accounts configured for automatic sponsored top-ups and a next automatic top-up date in the time range specified in the **pin_balance_transfer** utility's command-line parameters.

How BRM Handles Manual Standard Top-Ups

For manual standard top-ups, PCM_OP_PYMT_TOPUP does the following:

1. Depending on the payment method, does one of the following:
   - For *voucher* payment methods, calls PCM_OP_PYMT_POL_VALID_VOUCHER to perform these operations:
     - Call your voucher management system to validate the voucher and retrieve the balance impacts of the voucher's balances.
     - Determine whether the voucher has a currency balance, a noncurrency balance, or both.
     - Use the balance with the earliest validity start date and the balance with the latest validity end date to determine the validity period of the voucher.
     - Call PCM_OP_VOUCHER_ASSOCIATE_VOUCHER to associate the voucher with the account.
     - Return the preceding information and the voucher's balance impacts to PCM_OP_PYMT_TOPUP.
   - For *credit card* and *direct debit* payment methods, collects payment from the credit card agency or direct debit company, and then updates the specified balance.
   - For *cash* and *check* payment methods, receives the payment amount from the client application.
2. Calls PCM_OP_ACT_USAGE to generate an **/event/billing/topup** event.
3. If there are any **/item/loan_fee** and **/item/loan_debit** items for the account:

a. Transfers the corresponding amount from the top-up to pay off the loan. If the amount in the top-up is less than the amount due for the loan, one of the following happens:
   - The percent configured in the **LoanRepaymentPercent** business parameter is transferred and any remaining percent is credited to the main account balance.
   - If the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN has been customized for this scenario, the top-up is rejected and the customer is notified.

b. Calls PCM_OP_BILL_SET_LIMIT_AND_CR to subtract the amount from the balance group's PIN_FLD_OUTSTANDING_AMOUNT field.

c. Calls PCM_OP_CUST_MODIFY_PROFILE to subtract the amount from the loan profile's PIN_FLD_OUTSTANDING_AMOUNT field.

4. If topping up a noncurrency balance, does the following:
   - Calls PCM_OP_PYMT_POL_TOPUP_SET_VALIDITY to perform any customization on the validity dates for the noncurrency balance. By default, it does nothing.
   - Calls PCM_OP_BILL_DEBIT to apply a credit to the account's noncurrency sub-balance.

> **Note:**
>
> The cash and check payment methods do not support noncurrency balance top-ups.

5. If topping up a currency balance, calls PCM_OP_PYMT_COLLECT to apply a credit to the account's currency sub-balance.
6. Applies any top-up discount incentives to the account by calling PCM_OP_PYMT_POL_PURCHASE_DEAL.

How BRM Handles Automatic Standard Top-Ups

PCM_OP_PYMT_TOPUP performs automatic standard top-ups as follows:

1. Collects payment from the credit card agency or direct debit company, and then updates the specified balance.

> **Note:**
>
> Automatic standard top-ups support only the credit card and debit card payment methods.

2. Calls PCM_OP_ACT_USAGE to generate an **/event/billing/topup** event.
3. Retrieves the top-up amount from the specified **/topup** object.
4. Verifies that the top-up amount will not cause the *sum* of all automatic standard top-ups received during the current accounting cycle to exceed the account's automatic standard top-up cap.
5. If there are any **/item/loan_fee** and **/item/loan_debit** items for the account:
   a. Transfers the corresponding amount from the top-up to pay off the loan. If the amount in the top-up is less than the amount due for the loan, one of the following happens:

- The percent configured in the **LoanRepaymentPercent** business parameter is transferred and any remaining percent is credited to the main account balance.
- If the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN has been customized for this scenario, the top-up is rejected and the customer is notified.

b. Calls PCM_OP_BILL_SET_LIMIT_AND_CR to subtract the amount from the balance group's PIN_FLD_OUTSTANDING_AMOUNT field.

c. Calls PCM_OP_CUST_MODIFY_PROFILE to subtract the amount from the loan profile's PIN_FLD_OUTSTANDING_AMOUNT field.

6. If topping up a noncurrency balance, does the following:
   - Calls PCM_OP_PYMT_POL_TOPUP_SET_VALIDITY to perform any customization on the validity dates for the noncurrency balance. By default, it does nothing.
   - Calls PCM_OP_BILL_DEBIT to apply a credit to the account's noncurrency sub-balance.
7. If topping up a currency balance, calls PCM_OP_PYMT_COLLECT to apply a credit to the account's currency sub-balance.
8. Applies any top-up discount incentives to the account by calling PCM_OP_PYMT_POL_PURCHASE_DEAL.

How BRM Handles Recurring Standard Top-Ups

PCM_OP_PYMT_TOPUP performs recurring standard top-ups as follows:

1. Depending on the payment method, does one of the following:
   - For *voucher* payment methods, calls PCM_OP_PYMT_POL_VALID_VOUCHER to perform these operations:
     - Call your voucher management system to validate the voucher and retrieve the balance impacts of the voucher's balances.
     - Determine whether the voucher has a currency balance, a noncurrency balance, or both.
     - Use the balance with the earliest validity start date and the balance with the latest validity end date to determine the validity period of the voucher.
     - Call PCM_OP_VOUCHER_ASSOCIATE_VOUCHER to associate the voucher with the account.
     - Return the preceding information and the voucher's balance impacts to PCM_OP_PYMT_TOPUP.
   - For *credit card* and *direct debit* payment methods, collects payment from the credit card agency or direct debit company, and then updates the specified balance.
2. Calls PCM_OP_ACT_USAGE to generate an **/event/billing/topup** event.
3. Retrieves the top-up amount and resource ID from the specified **/topup** object.
4. Verifies that the number of recurring top-ups performed (in the **/topup** object's PIN_FLD_NUM_TOPUPS_DONE field) is less than the maximum number of recurring top-ups allowed (in the **/topup** object's PIN_FLD_NUM_TOPUPS field).
   After PIN_FLD_NUM_TOPUPS_DONE equals the value set in PIN_FLD_NUM_TOPUPS, no further recurring top-ups are performed.
5. If there are any **/item/loan_fee** and **/item/loan_debit** items for the account:

a. Transfers the corresponding amount from the top-up to pay off the loan. If the amount in the top-up is less than the amount due for the loan, one of the following happens:
   - The percent configured in the **LoanRepaymentPercent** business parameter is transferred and any remaining percent is credited to the main account balance.
   - If the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN has been customized for this scenario, the top-up is rejected and the customer is notified.

b. Calls PCM_OP_BILL_SET_LIMIT_AND_CR to subtract the amount from the balance group's PIN_FLD_OUTSTANDING_AMOUNT field.

c. Calls PCM_OP_CUST_MODIFY_PROFILE to subtract the amount from the loan profile's PIN_FLD_OUTSTANDING_AMOUNT field.

6. If topping up a noncurrency balance, does the following:
   - Calls PCM_OP_PYMT_POL_TOPUP_SET_VALIDITY to perform any customization on the validity dates for the noncurrency balance. By default, it does nothing.
   - Calls PCM_OP_BILL_DEBIT to apply a credit to the account's noncurrency sub-balance.
7. If topping up a currency balance, calls PCM_OP_PYMT_COLLECT to apply a credit to the account's currency sub-balance.
8. Applies any top-up discount incentives to the account by calling PCM_OP_PYMT_POL_PURCHASE_DEAL.
9. Calls PCM_OP_SUBSCRIPTION_CALCULATE_VALIDITY to calculate the due date for the next recurring top-up, which is set in the **/topup** object's PIN_FLD_NEXT_TOPUP_T field. It also increments the number of recurring top-ups completed by one in the **/topup** object's PIN_FLD_NUM_TOPUPS_DONE field.
10. Calls PCM_OP_PYMT_POL_POST_TOPUP to perform any customizations to the due date for the next recurring top-up. By default, this policy opcode does nothing.

How BRM Handles Manual Sponsored Top-Ups

PCM_OP_PYMT_TOPUP performs manual sponsored top-ups as follows:

1. Receives the top-up amount from a client application.
2. Verifies the following:
   - The status of the member is active.
   - **(Member-initiated top-ups only)** The top-up PIN is valid.
   - The top-up amount will not cause the total amount of credit charged to the owner account's balance group to exceed the credit limit of the associated balance in that balance group.
   - The top-up amount will not cause the *sum* of all top-ups received during the current accounting cycle of the owner account to exceed the group's top-up cap.
3. Performs these operations:
   - Calls PCM_OP_BILL_TRANSFER_BALANCE to transfer the top-up balances from the paying balance group to the receiving balance group.
   - Passes the reason ID and the reason domain ID used to differentiate sponsored top-up adjustments from other types of adjustments to PCM_OP_BILL_TRANSFER_BALANCE, which passes them to PCM_OP_AR_ACCOUNT_ADJUSTMENT.

- Updates the sum of all sponsored top-ups credited to members of the group during the group owner account's current accounting cycle. This value is stored in the PIN_FLD_CYCLE_TOPPED_AMT field of the LIMITS array in the **/group/topup** object.
  If the last sponsored top-up occurred in the owner account's *current* accounting cycle, PCM_OP_PYMT_TOPUP adds the amount of the current top-up to the value already in this field.

  If the last sponsored top-up occurred in the owner account's *previous* accounting cycle, the opcode sets this field to the amount of the current top-up.

4. If there are any **/item/loan_fee** and **/item/loan_debit** items for the account:
   a. Transfers the corresponding amount from the top-up to pay off the loan. If the amount in the top-up is less than the amount due for the loan, one of the following happens:
      - The percent configured in the **LoanRepaymentPercent** business parameter is transferred and any remaining percent is credited to the main account balance.
      - If the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN has been customized for this scenario, the top-up is rejected and the customer is notified.

   b. Calls PCM_OP_BILL_SET_LIMIT_AND_CR to subtract the amount from the balance group's PIN_FLD_OUTSTANDING_AMOUNT field.

   c. Calls PCM_OP_CUST_MODIFY_PROFILE to subtract the amount from the loan profile's PIN_FLD_OUTSTANDING_AMOUNT field.
5. Applies any top-up discount incentives to the account by calling PCM_OP_PYMT_POL_PURCHASE_DEAL.

How BRM Handles Automatic Sponsored Top-Ups

PCM_OP_PYMT_TOPUP performs automatic sponsored top-ups as follows:

1. Retrieves the top-up amount from the specified **/group/topup** object.
2. Verifies the following:
   - The status of the member is active.
   - The top-up amount will not cause the total amount of credit charged to the owner account's balance group to exceed the credit limit of the associated balance in that balance group.
   - The top-up amount will not cause the *sum* of all top-ups received during the current accounting cycle of the owner account to exceed the group's top-up cap.
3. Performs these operations:
   - Calls PCM_OP_BILL_TRANSFER_BALANCE to transfer the top-up balances from the paying balance group to the receiving balance group.
   - Passes the reason ID and the reason domain ID used to differentiate sponsored top-up adjustments from other types of adjustments to PCM_OP_BILL_TRANSFER_BALANCE, which passes them to PCM_OP_AR_ACCOUNT_ADJUSTMENT.
   - Updates the time that the member's last automatic sponsored top-up occurred to the current time.
     This value is stored in the PIN_FLD_LAST_TOPUP_T field of the LIMITS array in the **/group/topup** object. PCM_OP_PYMT_TOPUP uses this value to determine when to run the member's next automatic sponsored top-up.

- Calculates the time that the member's next automatic sponsored top-up will occur. This value is stored in the PIN_FLD_NEXT_TOPUP_T field of the LIMITS array in the **/group/topup** object.
- Updates the sum of all sponsored top-ups credited to members of the group during the group owner account's current accounting cycle. This value is stored in the PIN_FLD_CYCLE_TOPPED_AMT field of the LIMITS array in the **/group/topup** object.
  If the last sponsored top-up occurred in the owner account's *current* accounting cycle, PCM_OP_PYMT_TOPUP adds the amount of the current top-up to the value already in this field.

  If the last sponsored top-up occurred in the owner account's *previous* accounting cycle, the opcode sets this field to the amount of the current top-up.

4. If there are any **/item/loan_fee** and **/item/loan_debit** items for the account:
   a. Transfers the corresponding amount from the top-up to pay off the loan. If the amount in the top-up is less than the amount due for the loan, one of the following happens:
      - The percent configured in the **LoanRepaymentPercent** business parameter is transferred and any remaining percent is credited to the main account balance.
      - If the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN has been customized for this scenario, the top-up is rejected and the customer is notified.
   b. Calls PCM_OP_BILL_SET_LIMIT_AND_CR to subtract the amount from the balance group's PIN_FLD_OUTSTANDING_AMOUNT field.
   c. Calls PCM_OP_CUST_MODIFY_PROFILE to subtract the amount from the loan profile's PIN_FLD_OUTSTANDING_AMOUNT field.
5. Applies any top-up discount incentives to the account by calling PCM_OP_PYMT_POL_PURCHASE_DEAL.

Implementing Top-Ups in Custom Client Applications

All top-ups are performed by PCM_OP_PYMT_TOPUP. See the following topics:

- Implementing Manual Standard Top-Ups
- Implementing Automatic Standard Top-Ups
- Implementing Recurring Standard Top-Ups
- Implementing Manual Sponsored Top-Ups
- Implementing Automatic Sponsored Top-Ups
- Viewing Sponsored Top-Up History
- About Transferring Sponsored Top-Ups from Debit Balances

Implementing Manual Standard Top-Ups

To implement the manual standard top-up feature, configure your custom client application to accept the following top-up information and pass it in the input flist to the PCM_OP_PYMT_TOPUP opcode:

- For voucher top-ups, pass the following information in the PIN_FLD_VOUCHERS_INFO array:
  - Voucher serial number
  - Voucher PIN number

  - Bill unit (**/billinfo** object) to top up
  - Balance group to top up (optional)
  - Amount to top up (also include the resource ID for noncurrency resources)
  - Service to top up

> **Note:**
>
> To apply the voucher top-up to a *service-level balance group*, you must pass the **/service** object POID in the PIN_FLD_SERVICE_OBJ field of the PIN_FLD_VOUCHERS_INFO array.

- For credit card top-ups, pass the following information in the PIN_FLD_TOPUP_INFO array:
  - Bill unit to top up
  - Balance group to top up (optional)
  - Amount to top up (also include the resource ID for noncurrency resources)
  - Credit card number
  - Credit card expiration date
  - Credit card owner's name and address information
- For direct debit top-ups, pass the following information in the PIN_FLD_TOPUP_INFO array:
  - Bill unit to top up
  - Balance group to top up (optional)
  - Amount to top up (also include the resource ID for noncurrency resources)
  - Bank routing number
  - Bank account number
  - Direct debit owner's name and address information

Implementing Automatic Standard Top-Ups

To implement the automatic standard top-up feature:

1. In PDC or Pricing Center, set a credit floor and credit threshold for any package that contains bundles whose service balances you want your customers to top up. For example, to trigger an automatic standard top-up when an account balance falls below $30, set the credit floor to **-100** and the threshold to **70%**.
2. Configure your custom client application to accept the following top-up information:
   - **Payment method**

     To specify the payment method for automatic standard top-ups, set the appropriate value in the PIN_FLD_PAYINFO field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist (see step 3 for opcode names).
   - **Automatic top-up amount**

To specify the payment amount of each automatic standard top-up, set the appropriate value in the PIN_FLD_TOPUP_AMT field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

- **Automatic top-up cap**

  To specify the aggregate amount of automatic standard top-ups that the account can receive during an accounting cycle, set the appropriate value in the PIN_FLD_TOPUP_CAP field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

- **Bill unit to top up**

  To specify the bill unit that contains the balance to top up, set the appropriate value in the PIN_FLD_BILLINFO field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

3. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

> **Note:**
>
> Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.

For more information, see "Setting Up Top-Up Information in an Account".

Implementing Recurring Standard Top-Ups

To implement the recurring standard top-up feature:

1. Configure your custom client application to accept the following top-up information:
   - **Type of top-up**

     Set the type of top-up in the PIN_FLD_TYPE field under the PIN_FLD_TOPUP_INFO array to automatic (**0**), one-time (**1**), or recurring (**2**) of the called opcode's input flist.

   - **Top-up amount**

     Set the amount to top-up in the PIN_FLD_TOPUP_AMT field under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

   - **Maximum number of recurring top-ups**

     Set the maximum number of recurring top-ups that can be made in the PIN_FLD_NUM_TOPUPS field under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

   - **Interval between recurring top-ups**

     Set the interval between each recurring top-up in the following fields under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist:

     – PIN_FLD_OFFSET_UNIT specifies the interval's unit: weeks (**3**), days (**4**), or month (**5**).

     – PIN_FLD_NUM_UNITS specifies the interval's value.

- **Currency or noncurrency resource to top-up**

  Set the currency or noncurrency resource to top-up in the PIN_FLD_RESOURCE_ID field under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

- **Payment method**

  Set the payment method for recurring standard top-ups in the PIN_FLD_PAYINFO_OBJ field under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

- **Balance group to top-up**

  Set the balance group to top-up in the PIN_FLD_BAL_GRP_OBJ field under the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.

2. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

   > **Note:**
   >
   > Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.

3. Run the **pin_balance_transfer** utility on a regular basis:

   ```
   pin_balance_transfer -standard
   ```

For more information, see "Setting Up Top-Up Information in an Account".

Implementing Manual Sponsored Top-Ups

To implement the manual sponsored top-up feature:

1. Configure your custom client application to accept the following top-up information:
   - The POID of the account that *initiates* the creation or modification of the top-up configuration. Set this value in the level-one PIN_FLD_POID field of the called opcode's input flist (see step 2 for opcode names).
   - The POID of the account that will receive the top-ups (the member account). Set this value in the PIN_FLD_ACCOUNT_OBJ field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.
   - The POID of the **/topup** object associated with the account that will receive the top-ups. If the account does not have a **/topup** object, specify the POID type. Set this value in the PIN_FLD_POID field in the PIN_FLD_TOPUP_INFO array of the called opcode's input flist.
   - **Group to add member to**

     To specify the sponsored top-up group to add the member account to, set the POID of the appropriate **/group/topup** object in the PIN_FLD_POID field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode's input flist.

     If the object does not exist, pass the POID type.

   - **Group name**

To specify the name of the sponsored top-up group, set the appropriate value in the PIN_FLD_NAME field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode's input flist.

If you do not specify a group name, the group will be named **default**.

- **Group owner**

  To specify the sponsored top-up group owner account, set the POID of the owner account in the PIN_FLD_PARENT field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode's input flist.

- **Paying balance group**

  To specify the owner account's balance group to transfer top-up assets from, set the POID of the balance group in the PIN_FLD_BAL_GRP_OBJ field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode's input flist.

- **Balances to top up**

  To specify the balances to be topped up in the member account, set the IDs of the appropriate currency and noncurrency balance elements in the PIN_FLD_RESOURCE_ID fields in the PIN_FLD_GROUP_TOPUP_LIMITS array of the called opcode's input flist.

- **Balance top-up cap**

  To specify the maximum amount of a balance that the owner can transfer to its members during the owner's accounting cycle, set the appropriate value in the balance's PIN_FLD_TOPUP_CAP field in the PIN_FLD_GROUP_TOPUP_LIMITS array.

  > **Note:**
  >
  > This cap applies to the *sum* of all top-ups in the group, not to an individual member's top-ups. The cap should not exceed the credit limit of the paying balance group.

- **Receiving balance group**

  To specify the member account's balance group to transfer sponsored top-ups to, set the POID of the balance group in the member's PIN_FLD_BAL_GRP_OBJ field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the called opcode's input flist.

- **Top-up PIN**

  To specify the member's sponsored top-up PIN, set the appropriate value in the PIN_FLD_PIN field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the called opcode's input flist.

- **Top-up status**

  To specify the member's sponsored top-up status, set the appropriate value in the PIN_FLD_STATUS field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the called opcode's input flist.

2. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

> **Note:**
>
> Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.

Implementing Automatic Sponsored Top-Ups

To implement the automatic sponsored top-up feature:

1. Configure your custom client application to accept the following top-up information:
   - All top-up information listed in "Implementing Manual Sponsored Top-Ups".
   - **Automatic top-up amount**

     To specify the amount of a balance to transfer from the owner to the member during each automatic sponsored top-up, set the appropriate value in the balance's PIN_FLD_TOPUP_AMT field in the PIN_FLD_GROUP_TOPUP_LIMITS array.

     This amount applies to every member in the group.
   - **Automatic top-up frequency**

     To specify the number of days in the member's automatic sponsored top-up cycle, set the appropriate value in the member's PIN_FLD_TOPUP_INTERVAL field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the called opcode's input flist (see step 2 for opcode names).

     This interval applies only to the member account.
2. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

   > **Note:**
   >
   > Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.
3. Run the **pin_balance_transfer** utility on a regular basis:

   ```
   pin_balance_transfer [-start mm/dd/yy] [-end mm/dd/yy]
   ```

Viewing Sponsored Top-Up History

To display information about sponsored top-up transactions, configure your custom client application to call PCM_OP_PYMT_FIND_TOPUP_EVENTS to retrieve balance transfer events (**/event/billing/adjustment/account** objects) in which top-up transactions are recorded.

Historic sponsored top-up information can be displayed for both sponsored top-up owner accounts and member accounts:

- Member accounts can view only the sponsored top-ups that they received.
- Owner accounts can view all sponsored top-ups associated with their sponsored top-up groups.

To retrieve all sponsored top-up events associated with a group owner account or a member account, include the following values in the PCM_OP_PYMT_FIND_TOPUP_EVENTS input flist:

- Account's POID in the PIN_FLD_POID field

> **Note:**
>
> This field stores the POID of the account that *initiates* the search. If a POID type rather than an actual POID is provided, an actual balance group POID must be set in the flist's PIN_FLD_BAL_GRP_OBJ field.

- No value in these fields:
  - PIN_FLD_BAL_GRP_OBJ

> **Note:**
>
> If no account POID is provided, a balance group POID must be provided. In such cases, the opcode retrieves top-up events associated only with the specified balance group.

  - PIN_FLD_REASON_ID
  - PIN_FLD_REASON_DOMAIN_ID

If an owner account has multiple sponsored top-up groups, retrieve sponsored top-up events related only to one group by including the following values in the PCM_OP_PYMT_FIND_TOPUP_EVENTS input flist:

- Group owner account's POID in the PIN_FLD_POID field
- Paying balance group's POID in the PIN_FLD_BAL_GRP_OBJ field
- No value in these fields:
  - PIN_FLD_REASON_ID
  - PIN_FLD_REASON_DOMAIN_ID

By default, PCM_OP_PYMT_FIND_TOPUP_EVENTS retrieves all sponsored top-up debit and credit adjustment events associated with the initiating account. Thus, if a sponsored top-up group owner account initiates the search, the opcode retrieves two events for each top-up:

- An event for debiting the top-up amount from the owner's paying balance group
- An event for crediting the top-up amount to a member's receiving balance group

To limit the search to debit *or* credit adjustment events, include the appropriate reason ID and reason domain ID in the PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID fields of the opcode's input flist.

> **Note:**
>
> If you create custom reason and reason domain IDs for sponsored top-up events, PCM_OP_PYMT_FIND_TOPUP_EVENTS returns events associated with *all* the IDs unless you limit the search to specified IDs.

About Transferring Sponsored Top-Ups from Debit Balances

To perform a sponsored top-up, PCM_OP_BILL_TRANSFER_BALANCE must transfer assets from a debit balance in a group owner account to a debit or credit balance in a member account. By default, however, this opcode transfers assets only from credit balances. To enable the opcode to transfer assets from a debit balance, the PIN_FLD_VERIFY_BALANCE field in its input flist is set to **PIN_BOOLEAN_FALSE** by PCM_OP_PYMT_TOPUP.

> **Note:**
>
> If this field is not set (default) or is set to **PIN_BOOLEAN_TRUE**, the opcode cannot transfer assets from debit balances.

Setting an Account's Sponsored Top-Up Member Status and PIN

If an account is a member of a sponsored top-up group, its group status and top-up PIN are set as follows:

- When an owner account initiates adding a member to the group, the member's group status and PIN are set according to the information in the PCM_OP_CUST_SET_TOPUP input flist. If member status is not specified in the flist, it is set to **active**.
- When a member account initiates adding itself to a group, the member's group status is set to **inactive** and the PIN is set to **NULL** no matter what values are provided for those items in the input flist. After the member is added to the group, the group owner must activate the member's member and sets its top-up PIN.
- When a member account initiates a modification of its sponsored top-up settings after being added to a group, its group status is reset to **inactive**. This occurs even if the modification is not related to member status. The group owner must then reactivate the member.

Activating Sponsored Top-Up Group Members

To receive sponsored top-ups, a member's group status must be *active*. By default, only groups owners can activate a member. To enable members to activate themselves, customize the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.

To activate members whose group status is *inactive* or *closed*:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member's PIN_FLD_STATUS field in the MEMBERS array of the opcode's input flist to the value associated with the **PIN_STATUS_ACTIVE** status in the *BRM_home***/include/ops/pcm.h** header file.

Inactivating Sponsored Top-Up Group Members

A member whose group status is inactive can use any outstanding topped-up credit in its topped-up balance group, but it cannot receive any more top-ups from the group until its member status is reactivated. In addition, it cannot join another sponsored top-up group.

To inactivate a member's group status:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member's PIN_FLD_STATUS field in the MEMBERS array of the opcode's input flist to the value associated with the **PIN_STATUS_INACTIVE** status in the *BRM_home***/include/ops/pcm.h** header file.

> **Note:**
>
> You can also inactivate sponsored top-ups by changing the status of the member *account* to inactive. To change the status of an account, see "Changing Account and Service Status" in *BRM Managing Customers*.

To inactivate the group status of every member in a group, change the status of the sponsored top-up group owner *account* to inactive.

When a group owner account is inactive, the members of its sponsored top-up groups can use any outstanding topped-up credit in their topped-up balance groups, but they cannot receive any more top-ups from the group until the owner account is reactivated, and they cannot join another sponsored top-up group.

Setting Sponsored Top-Up Member PINs

By default, only groups owners can set a member's PIN.

To assign a top-up PIN to a member:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member's PIN_FLD_PIN field in the MEMBERS array of the opcode's input flist to the appropriate string value.

> **Note:**
>
> - By default, top-up PINs do not have to be unique. Members in the same group and in different groups can have the same top-up PIN.
> - To enable members to set their own PINs, customize the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.

Offering Discount Incentives with Top-Ups

You can offer customers discount incentives whenever they top up an account balance by a specified amount. For example, you can offer 20 free peak minutes whenever a customer makes a $50 top-up.

To offer top-up discount incentives:

1. Set up the discount in PDC or Pricing Center.
2. Configure the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode to validate the discount.

   By default, this policy opcode is an empty hook provided to facilitate customization. You can customize it to apply discounts to account balances when they are topped up. For example, this policy opcode might grant 60 minutes of usage for every $50 top-up.

   PCM_OP_PYMT_POL_PURCHASE_DEAL is called by PCM_OP_PYMT_TOPUP.

Suspending and Recycling Payments

This section describes the opcodes used for payment suspense management.

How BRM Tracks Suspended Payments

All payments and payment reversals contain a transaction ID (TRANS_ID field), which is a unique identifier that enables you to track each payment and reversal in BRM:

- For *payments*, the transaction ID identifies the payment transaction from the third-party payment processor. If a payment is submitted to BRM without a transaction ID, one is assigned to it.
- For *reversals*, the transaction ID is generated by BRM.

In addition to having a transaction ID, payments and reversals have another ID that is used to track all actions performed on a specific payment: a subtransaction ID (SUB_TRANS_ID field) for payments, and a paytransaction ID (PAYMENT_TRANS_ID field) for reversals. These ID values are used to find all payments and reversals that occur due to the recycling of an original payment.

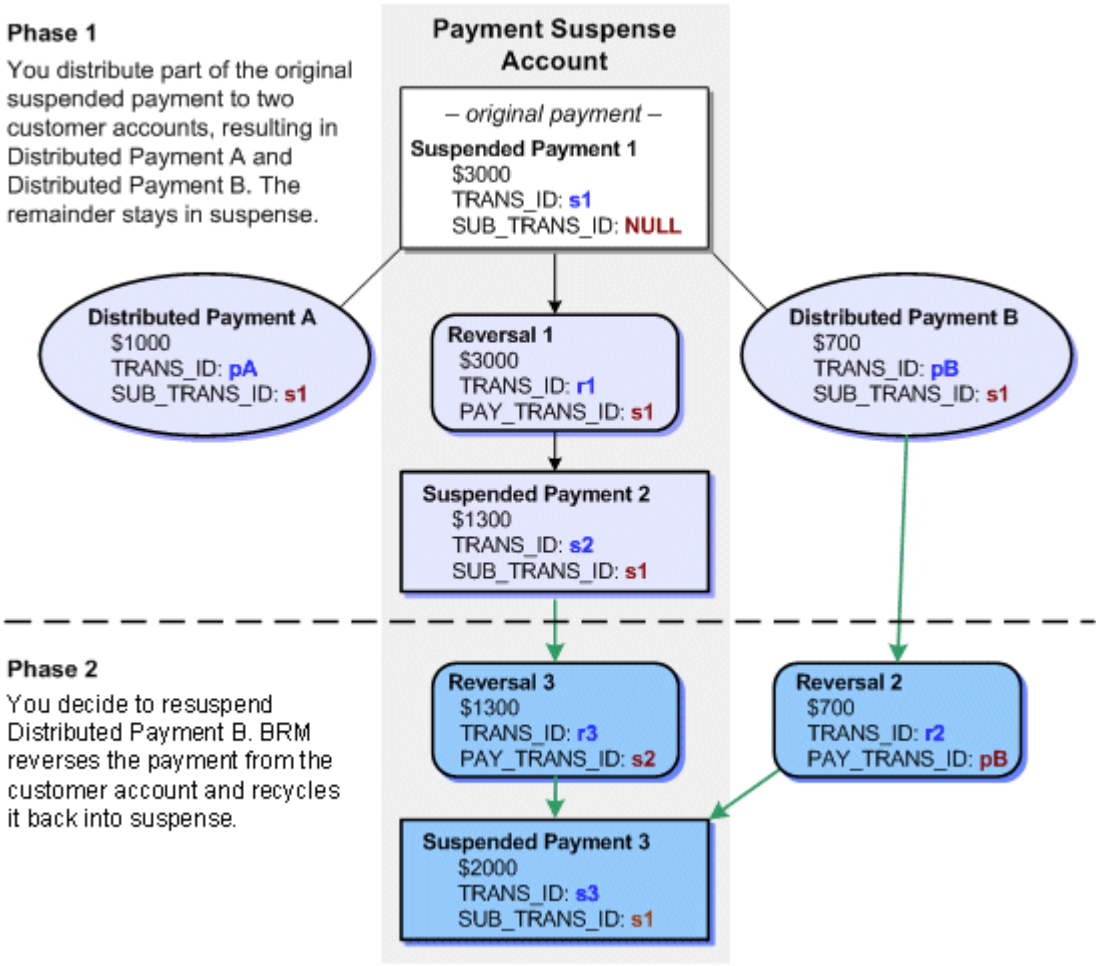
- The SUB_TRANS_ID value of an original (never-recycled) payment is **NULL**.
- The SUB_TRANS_ID value of a recycled payment is equal to the transaction ID of its original payment.
- The PAYMENT_TRANS_ID value of a payment reversal is equal to the transaction ID of the payment it reversed.

In the example shown in Figure 21-1, a $3000 payment is suspended when it first arrives in BRM so that it can be posted to individual accounts within the corporation. The original payment is represented in white.

The payment analyst first partially distributes the original suspended payment to two accounts. The first distributed payment is for $1000 to Account A and the second is for $700 to Account B. The remaining $1300 is resuspended. The results of this operation are represented in Phase 1.

Later, the payment analyst realizes that the distributed payment made to Account B was erroneous and puts the $700 payment back into suspense, leaving $1000 in Account A and $2000 in the suspense account. The results of this operation are represented in Phase 2.

Figure 21-1 Suspended Payment Distribution Example



In this figure, all payments except the original payment are recycled payments. Notice that:

- Both distributed payments have the same subtransaction IDs (**s1**). The SUB_TRANS_ID value is the same as the TRANS_ID value of the original payment.
- All of the recycled payments have SUB_TRANS_ID values that match the original payment's TRANS_ID values.
- Each reversal has a PAYMENT_TRANS_ID value that is equal to the TRANS_ID value of the payment it reversed.

When an original suspended payment is applied to a customer account, the following operations occur:

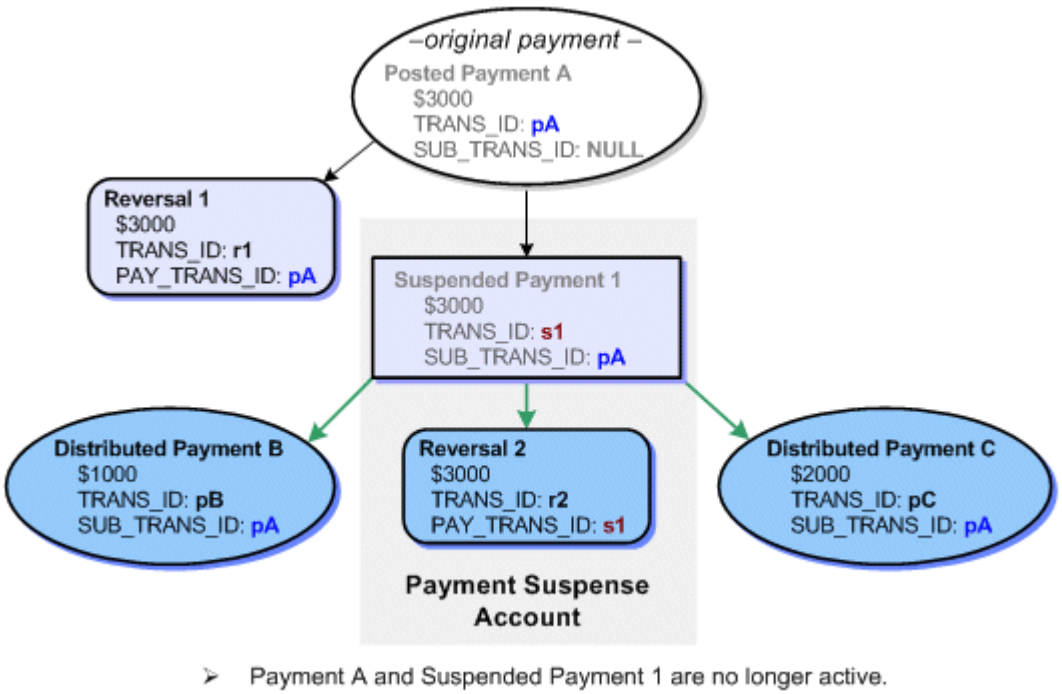
1. The suspended payment is reversed, and an **/event/billing/reversal** object is created. The reversal object's PAYMENT_TRANS_ID value is equal to the TRANS_ID value of the suspended payment.
2. A new payment (**event/billing/payment/***pay_type* object) is applied to the customer account and contains the following information:
   - The original suspended payment's details, including a value for any amount remaining in suspense.
   - A SUB_TRANS_ID value that is equal to the suspended payment's TRANS_ID value.

3. If an amount remains in suspense, a new suspended payment is created for that amount; the original payment is not adjusted with a new amount. Instead, the new suspended payment receives a SUB_TRANS_ID value that is equal to the original payment's TRANS_ID value.

   Likewise, when a suspended payment is partially allocated to one or more accounts, and then an allocated payment is resuspended, the current suspended payment that was partially allocated is reversed and a new suspended payment containing the new amount is created.

> **Note:**
>
> An original payment does not have to be a suspended payment. For example, if a payment was posted successfully to a customer account when it was received by BRM, it is an original payment. If it is later suspended and then reposted to customer accounts, both the suspended payment and the posted payment will have SUB_TRANS_ID values that match the TRANS_ID value of the original successful payment as shown in Figure 21-2.

Figure 21-2 SUB_TRANS_ID of a Suspended Payment



Customizing Payment Suspense

You can customize payment suspense in several ways. See the following topics:

- Customizing Payment Suspense Validation
- Customizing Suspended Payment Distribution

- Customizing Payment Failure Reason Codes
- Handling Custom Payment Methods

Customizing Payment Suspense Validation

The PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode validates payments to determine whether they can be successfully posted or whether a failed, unconfirmed payment needs reversal. If automatic write-off reversals are enabled, it also determines whether BRM should perform a write-off reversal.

In default implementations, BRM considers two questions when determining whether a payment should be suspended:

- Is the account number present and valid?
- Is the account closed?

By default, the bill number is not initially used to suspend payments. It is used only when the account number is missing.

A payment is suspended in the following situations:

- The account is closed.
- The account number is missing and the bill number is missing.
- The account number and the bill number are invalid.
- The account number does not match the account number from the bill.

A payment is posted if the account is not closed and the following is true:

- The account number is present and valid. If the bill number is missing, the payment is posted at the account level.
- The account number is missing but the bill number is present and valid. Such payments are posted at the bill level.

You can broaden this scope by customizing the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to perform validation using additional criteria (for example, customer segment, payment amount, and so on). BRM uses these criteria to determine whether the payment validation logic should mark a payment for suspense.

> **Note:**
>
> If you include additional criteria in the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode that is not already part of the objects created by payment processing, you must also extend these objects by adding these fields so that the object records all the criteria considered by payment validation.

If neither the bill number nor bill POID was submitted with the payment, you can configure BRM to find the bill based on the due amount.

See "Finding Bills by Due Amount" in *BRM Collecting Payments* for more information.

Customization Example: Suspending Large Payments

As an example, if you want a payment analyst to always examine suspiciously large cash payments or cash payments that appear to arrive from the Internet, you can customize BRM to

suspend any payments that meet these conditions. If a cash payment is suspiciously large, there may be a customer error or recording error that must be investigated. Because the Internet is not a likely source of cash payments, you may want to obtain extra confirmation on how this payment was made.

This type of customization includes the following tasks:

- Modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to include logic that checks the PIN_FLD_PAYMENT_TYPE field to determine if this is a cash payment and also checks PIN_FLD_CHANNEL_ID to see whether the payment was made over the Internet. If both conditions are met, have the opcode set PIN_FLD_STATUS to PIN_PYMT_SUSPENSE.
- Also modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to include logic that checks the PIN_FLD_AMOUNT field to see whether the payment amount is greater than the amount you establish as the threshold for suspiciously large payments (for example, $10,000). If this is a cash payment and the payment amount exceeds the threshold, have the opcode set PIN_FLD_STATUS to PIN_PYMT_SUSPENSE.

Customization Example: Threshold for Suspending Payments

You can customize the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to enforce business practices related to suspense. For example, if you find that having payment analysts review suspended payments for very small amounts costs you more than the payments are worth, you can customize BRM so that it does not move any payment of less than a certain amount into the payment suspense account.

To create a suspense threshold, you customize the opcode to check the PIN_FLD_AMOUNT field to see whether the payment amount is less than the threshold amount (for example, $1). You check this condition for any payment whose status is set to PIN_PYMT_SUSPENSE. For each payment that meets these conditions, you set the account POID to a special account you set up for this type of unallocatable payment. You also set PIN_FLD_STATUS to PIN_PYMT_SUCCESSFUL so that PCM_OP_PYMT_COLLECT can post the payment.

> **Note:**
>
> When you implement any of the customizations just discussed, you modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode. This opcode validates only externally initiated payments; it does not validate BRM-initiated payments. Therefore, none of the customizations you implement through this opcode affect BRM-initiated payments.

Customization Example: Finding Unconfirmed Payments

In the case of failed unconfirmed payment, if the payment processor is not able to send a transaction ID with each payment, you can modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to find the original unconfirmed payment by using other payment properties that are sent from the payment gateway, such as the original unconfirmed payment amount. When the original unconfirmed payment is found, the item object to which the payment was applied can be loaded into the input flist for the PCM_OP_PYMT_APPLY_FEES opcode.

> **Note:**
>
> If you used failed payment properties to locate the payment, you must also update the **/event/billing/payment/failed** object to contain the same properties that you are using to locate the original unconfirmed payment.

Customization Example: Error Handling

You can modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to handle errors that occur because the original unconfirmed payment cannot be found or the transaction IDs do not match.

Customizing Suspended Payment Distribution

The PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode enables you to define custom rules for handling payments that are being guided into suspense. For example, you can modify this policy opcode to track the number of times the same payment is guided back to suspense after being applied to the wrong customer account.

In addition, if your company processes distributed payments, you can implement rules to distribute a payment directly to the proper accounts, without first saving the payment to the suspense account. For example, by using the account number associated with the original payment, the original payment amount, and the account number associated with each subpayment amount, you can automatically divide the payment into distributed payments and allocate them to the designated accounts.

For multiple suspense accounts, the PCM_OP_PYMNT_POL_SUSPEND_PAYMENT policy opcode must be modified to select a payment suspense account that matches the criteria specified while creating the account. Otherwise, the policy opcode selects a payment suspense account that matches the login schema.

Customizing Payment Failure Reason Codes

To customize payment failure reason codes, use the PCM_OP_PYMT_POL_CHARGE policy opcode. This policy opcode provides the ability to map the online and offline payment result to the payment status and the reason IDs defined in the **/strings** object.

In the output flist PIN_FLD_PAYMENT_REASONS array, the array of PIN_FLD_REASON_ID fields contains the failure reasons sent by the payment processor. You can configure this policy opcode to apply fees for failed credit card and direct debit transactions based on the reason for failure.

The input flist contains a results array for a payment batch, including reasons for payment failures.

The output flist contains the payment method, transaction ID, and an array of reason IDs for failed payments.

Handling Custom Payment Methods

If you create custom payment methods for your BRM system, you must customize Payment Suspense Center to handle them. This overview procedure describes how to create custom storable classes and fields and enable Payment Center to handle them.

For detailed information on creating custom storable classes and fields, see Creating Client Applications by Using Java PCM and Creating Custom Fields and Storable Classes in *BRM Developer's Guide*.

1. Complete the following tasks by using Storable Class Editor:
   a. Create your storable classes and fields in the Java PCM package.
   b. Create source files for your custom fields.

> **Note:**
>
> Storable Class Editor creates a C header file called **cust_flds.h**, a Java properties file called **InfranetPropertiesAdditions.properties**, and a Java source file for each custom field.

2. In the directory where Storable Class Editor created the Java source files, compile the source files:

```
javac -d . *.java
```

3. Package the new storable class files into a JAR file. For example:

```
jar cvf customfields.jar *.class
```

4. Copy the contents of the **InfranetPropertiesAdditions.properties** file and paste it into the Payment Suspense Center **Infranet.properties** file. By default, this file is located in:

```
C:\Program Files\Portal Software\PymtSuspCenter\PaymentCenter
```

5. Append the location of the JAR file to the PAYCTRCP environment variable path. For example:

```
;.;C:\Program Files\Portal Software\PymtSuspCenter\customfields.jar;
```

About the Suspense Account

When BRM creates the payment suspense account, PCM_OP_CUST_POL_PREP_NAMEINFO first prepares the contact information for validation and then determines whether Payment Suspense Manager is enabled. If it is enabled and the account being checked is the payment suspense account, this opcode retrieves the POID of the **/config/psm** object if that object exists. BRM references the **/config/psm** object to locate the payment suspense account whenever it suspends a payment or you correct a payment.

The payment suspense account POID is then passed to PCM_OP_CUST_SET_NAMEINFO. This opcode stores the account POID in the **/config/psm** object.

Suspending Payments during Payment Processing

During payment processing, PCM_OP_PYMT_COLLECT calls PCM_OP_PYMT_VALIDATE_PAYMENT, to determine the status of the payment records. (For information about PCM_OP_PYMT_COLLECT, see "Collecting Payments".)

When PCM_OP_PYMT_VALIDATE_PAYMENT receives a payment to validate, it determines whether the payment should be suspended and prepares it for posting by enriching the flist with any missing information.

It begins by evaluating the input flist to determine whether it contains the channel ID for the payment. If not, it gets the channel ID from the **/config/ach** object and adds it to the flist.

Then, PCM_OP_PYMT_VALIDATE_PAYMENT calls PCM_OP_PYMT_POL_VALIDATE_PAYMENT to validate the payment and further enrich the flist with any missing payment information.

After PCM_OP_PYMT_VALIDATE_PAYMENT receives the results from PCM_OP_PYMT_POL_VALIDATE_PAYMENT, it returns the flist to PCM_OP_PYMT_COLLECT for use when the payment is posted.

Flags are not used directly by PCM_OP_PYMT_VALIDATE_PAYMENT. They are passed in from PCM_OP_PYMT_COLLECT for PCM_OP_PYMT_SELECT_ITEMS.

How Payments Are Recycled to and from Suspense

PCM_OP_PYMT_RECYCLE_PAYMENT removes a payment from a source account and posts it to a target account. It is called when one or more payments in Payment Center are submitted to the BRM database. This opcode is called when a single payment or a list of distributed payments is transferred between the payment suspense account and one or more customer accounts.

> **Note:**
>
> It is not recommended to have more than one payment suspense account in the **/config/psm** object.

For account payment to multiple bill units, there can be more than one event generated for an individual payment. Therefore, the output flist of PCM_OP_PYMT_RECYCLE_PAYMENT shows all the payment events.

When a payment is recycled from suspense to a customer account, the input flist generated for the submission includes the corrected account number, corrected bill number, and the transaction ID of the original payment. It can also contain other corrected information if you customized BRM to include additional validation criteria.

When a transaction is opened, PCM_OP_PYMT_RECYCLE_PAYMENT handles all of the activities required to move each payment in the CHARGES array from the source account to the target account. The transaction ID is recorded in all the event objects created when this opcode processes a payment.

PCM_OP_PYMT_RECYCLE_PAYMENT validates that the source account is the account to which the payment was posted and that the destination account is the payment suspense account. If both requirements are true, the payment is recycled to suspense.

The CHARGES array contains two fields that determine the direction of the transfer:

- PIN_FLD_EVENT_OBJ: This field identifies the payment event. The event, in turn, identifies the *source account*, the account that owns the payment. This is the account from which the opcode transfers payments.
- PIN_FLD_ACCOUNT_OBJ: This field identifies the *target account*, the account to which the opcode transfers payments.

For example, if the account referenced in PIN_FLD_EVENT_OBJ is the suspense account and the account referenced in PIN_FLD_ACCOUNT_OBJ is a customer account, the opcode distributes all or part of the suspended payment to the customer account.

If a payment is being moved from suspense, the PIN_FLD_EVENT_OBJ field establishes the payment suspense account as the source account, and the PIN_FLD_ACCOUNT_OBJ field

contains one or more POIDs of valid customer accounts. If a payment is being moved to suspense, the PIN_FLD_EVENT_OBJ field establishes a valid customer account as the source account, and the PIN_FLD_ACCOUNT_OBJ field contains the POID of the payment suspense account.

The source and target accounts, combined with the number of payments in the CHARGES array, determine the operation that PCM_OP_PYMT_RECYCLE_PAYMENT performs.

PCM_OP_PYMT_RECYCLE_PAYMENT performs the following operations when recycling payments:

1. Opens a transaction.
2. Determines the direction of the payment recycling: to or from the payment suspense account. The transfer direction is determined by the active payment's account POID and the target account POID.
   - Determines if multiple distributed payments are being handled.
   - Checks that the currency is the same for every charges element.
   - If it is returning an entire distribution to suspense, calls PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH to retrieve the active suspended payment information, including the amount remaining in the suspended payment after recycling, and adds it to the CHARGES array.
3. Chooses a course of action based on the source account/target account combination plus the information in the CHARGES array.
4. Calls PCM_OP_PYMT_COLLECT to apply the payments to the target accounts, prepares the payment batch, and enriches the PCM_OP_PYMT_COLLECT input flist with the following information:
   - The total charge amount.
   - Total reversal amount, for posted payments that are being suspended.
   - Total original payment amount, for payments that are being posted from the payment suspense account to a customer account.

   PCM_OP_PYMT_COLLECT, in turn, calls PCM_OP_PYMT_VALIDATE_PAYMENT to validate the information in the CHARGES array. If an invalid scenario exists, the payment is guided to suspense.

   > **Note:**
   >
   > For payments guided to suspense due to validation failures, PCM_OP_PYMT_RECYCLE_PAYMENT rolls back the entire transaction after PCM_OP_PYMT_COLLECT completes.

5. Prepares the reversal batch, creating new elements and setting a new reason code.

   If PCM_OP_PYMT_RECYCLE_PAYMENT is being called for a distribution and the total of the charges in the input flist does not equal the amount to be distributed, the opcode determines whether the input flist specifies an amount to be placed in suspense.
   - If the input flist does not specify an amount to be placed in suspense, the opcode posts the remaining amount to the suspense account.
   - If the input flist specifies an amount to be placed in suspense, the opcode generates an error.

6. Calls PCM_OP_PYMT_COLLECT to perform these tasks:
   - Create the payment batch.
   - Create a suspended payment if an amount remained in the suspended payment after it was applied to one or more accounts.
   - Update the **/event/billing/payment/***pay_type* object with the SUB_TRANS_ID value specified in the PIN_FLD_CHARGES array of the flist.
7. If the payment batch was successful, updates the PCM_OP_BILL_REVERSE input flist with the following information:
   - The reason code for the reversal
   - The PAYMENT_TRANS_ID value
8. Calls PCM_OP_BILL_REVERSE to create the reversal batch.

   PCM_OP_BILL_REVERSE updates the **/event/billing/reversal/***pay_type* object with the PAY_TRANS_ID, the POID of the payment suspense account, and the reversal total.

   If PCM_OP_PYMT_COLLECT fails or if any information is invalid, PCM_OP_PYMT_RECYCLE_PAYMENT rolls back the entire transaction, leaving the suspended payment in its original state.
9. Records the success or failure of the entire operation in the PIN_FLD_RESULTS field of the output flist.

Understanding Payment Recycling

Payment recycling is a transactional process consisting of two sequential operations:

1. Reversing a payment from a source account.
2. Applying a payment to a target account.

Payment recycling occurs each time you move a payment from a customer account to the payment suspense account or from the payment suspense account to a customer account. When a payment is recycled, the current payment is reversed, and all of the payment information, except for the original payment date, is transferred to the new recycled payment.

BRM uses the PIN_FLD_STATUS field of a payment to validate payments before they are recycled and submitted to an account. The status code is returned by Payment Center and can be configured.

Payments can be recycled any number of times; however, each recycled payment can be traced back to only one original payment. BRM uses the payment's transaction ID and subtransaction ID to track their origin and descendants.

About Original Payments

An original payment is the first instance of a payment that is saved to BRM through a payment processor. Original payments can be saved initially to the suspense account or to a customer account. The distinguishing characteristic of an original payment is that it has never been recycled.

Because an original payment is the first instance of a payment, it does not have a subtransaction ID, which is an ID used to trace a recycled payment back to its original payment.

An original payment can be the ancestor of one or more recycled payments, but a recycled payment can be traced back to only one original payment. Both suspended payments and those posted in customer accounts can be original payments.

About Payment Transfer Direction and Verification

When BRM receives a distribution list from Payment Center, it determines the direction of the payment transfer: from the payment suspense account to a customer account, or to the payment suspense account from a customer account.

If a payment is being moved from suspense, BRM establishes the payment suspense account as the source account, and one or more customer accounts as the destination. If a payment is being moved to suspense, BRM establishes a valid customer account as the source account, and the payment suspense account as the destination.

Before recycling a payment, BRM verifies that there are no conditions present that prevent the transfer. These conditions include:

- The currency for the source account and all target accounts is different.
- The source and target accounts are both customer accounts.
- Multiple suspended payments are recycled in the same operation.
- You attempt to partially suspend a payment originally posted to a customer account. You can only perform this action if you suspend the *entire* payment.
- You attempt to distribute a failed payment from the suspense account into a customer account.
- You attempt to return a distributed payment from a customer account to the suspense account, but the suspended payment is also a failed payment. This can happen when you distribute a suspended payment to customer accounts and, afterward, the payment fails for financial reasons (for example, the bank will not honor the check). In this case, you cannot return the distributed payment to suspense.

About Recycling Payments from Suspense

When you post a suspended payment to a customer account, the following operations occur if validation and transfer verification are successful:

1. The suspended payment is reversed (an **/event/billing/reversal** object is created).
2. The payment item for the suspended payment is closed.
3. A recycled payment is posted in the customer account. The payment status reflects that the payment was successfully recycled.
4. Bills, bill items, or both in the account might be closed, depending on the payment allocation level.
5. A new suspended payment is created for any unallocated amount (partially allocated payments).

You do not have to fully allocate a suspended payment. If you partially allocate a suspended payment and an amount remains in suspense, you can apply the remaining amount to any account at any time.

About Recycling Payments to Suspense

When you suspend a payment that has been posted in a customer account, the following operations occur if validation is successful:

1. The posted payment is reversed (an **/event/billing/reversal** object is created).
2. Any accounts receivable items (**/item/payment**) that were previously closed by the payment are reopened.
3. A recycled payment is created in the payment suspense account. The payment status reflects that the payment was successfully recycled.
4. A new billable item for the recycled suspended payment is created.

How Payment Reversals Work with Suspense and Recycling

Payments are reversed in BRM for a variety of reasons, the most common of which is that the funds for a payment are never actually deposited (for example, when a check does not clear). Payments also can be reversed as part of the suspense process.

When a payment is reversed, two things happen:

- Any bills or items previously closed by the payment are reopened.
- The payment is deactivated in the BRM system.

There are three types of reversals in BRM. The first two, indirect reversal and reversal to remove an unallocatable suspended payment, are related to the suspense process. The third, direct reversal, is not.

- **Reversals that occur indirectly during the recycling process**

  Indirect reversals occur when you transfer a suspended payment to a customer account or from a customer account to suspense. With an indirect reversal, the payment is removed from the source account and moved to the target account, resulting in a complete reversal of the payment in the source account so that the payment information can be transferred to the destination account as a recycled payment. If the payment being recycled had been posted in a customer account, any bills and items that were closed due to the payment are reopened.

  Indirect reversals are assigned a G/L ID of **113**, placing the payment amount in a special G/L bucket so that you can keep a separate record of these reversals.

- **Reversals that occur when you remove a payment as unallocatable**

  If a suspended payment cannot be fixed but you want to track revenue for these payments and get reports on how much unallocatable revenue you have, you can remove them from suspense as unallocatable. When you do this, BRM reverses the payment in the payment suspense account and assigns it to a special G/L ID bucket. These reversals are rare in BRM. Even though they are part of suspense, they occur outside of the recycling process.

- **Reversals that you perform directly by using a third-party application**

  Direct reversals occur when you use a third-party application to reverse a payment that was recorded in the BRM database but never actually deposited. Direct reversals are the most frequent type of reversal in BRM, and they occur outside of the suspense and recycling processes.

  Direct reversals reverse an active payment completely and reopen any bills and items so the payment can be made again. Unlike reversals that occur during recycling, direct reversals are not initiated by the creation of a new recycled payment.

About Directly Reversing Payments from BRM

Payment reversals are necessary when a payment is recorded in the BRM database but the payment is not deposited.

When you have Payment Suspense Manager enabled, payments may be reversed due to payment recycling, which is different from directly reversing the payment from the BRM database. Reversals that occur due to recycling happen automatically, and the funds are transferred from a source account to a target account: they are not removed completely from BRM.

The type of payment reversal discussed here is different from recycled payment reversals. Here, you manually reverse a payment to remove it from the system. A recycled payment reversal is a consequence of transferring a payment to a target account.

The following restrictions apply when reversing payments from BRM:

- You cannot directly reverse recycled payments: you can reverse only original (nonrecycled) payments with a SUB_TRANS_ID value of **NULL**. Therefore, when creating the payment reversal batch for a payment that entered the database as suspended, you must identify the original suspended payment rather than the active recycled payment. When the reversal batch is submitted to BRM, all active recycled payments that are descended from the original payment will be internally reversed.
- When directly reversing a payment, you can reverse only original payments. If you reverse an original suspended payment, BRM reverses all of the active recycled payments that were distributed to customer accounts from that payment.

Retrieving Recycled Payments

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH retrieves payment information such as the payment amount, transaction ID, subtransaction ID, account and bill number, payment types, and payment status. If you specify that more information be returned, this opcode can return additional payment information such as the action owner and the date suspended.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH is called by:

- Payment Center to provide additional information on the payments returned by a search (for example, allocated amount and unallocated amount).
- PCM_OP_BILL_REVERSE when it reverses a payment that was posted to the suspense account when it entered the BRM database. This opcode uses PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH to find all active payments associated with the original suspended payment so that those payments also can be reversed. For more information, see "How Suspended Payments Are Reversed".
- PCM_OP_PYMT_RECYCLE_PAYMENT when it returns an entire set of distributed payments to the suspense account.

As input, PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH receives the transaction ID of the payment it is to search for. It also receives an optional PIN_FLD_RESULTS array, which, with the PCM_OP_FLAG_READ_RESULT flag, determines the type of information returned by the opcode.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH performs the following operations when searching for payments:

1. Verifies that the transaction ID in the PIN_FLD_TRANS_ID field is for an original suspended payment. To do this, the opcode checks the PIN_FLD_SUB_TRANS_ID field. If this field is not **NULL**, the payment is not an original payment, and the opcode returns an error.
2. Searches for all payments whose subtransaction ID matches the PIN_FLD_TRANS_ID field of the payment in the input flist.

3. Checks to see whether the PIN_FLD_RESULTS array in the input flist and the PCM_OP_FLAG_READ_RESULT flag are present. The opcode returns payment information based on these presence of the flag and array, as follows:
   - If neither the flag nor the array is present, the opcode returns only the fields in the output flist.
   - If the flag is present, the opcode returns all the fields in the **/event/billing/payment** object.
   - If the flag is not present but the array is, the opcode returns the fields in specified in the PIN_FLD_RESULTS array plus the payment information fields normally included in the output flist.
   - If the both the flag and array are present, the opcode ignores the array and returns all the fields in the **/event/billing/payment** object.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH does not retrieve payment states for payments that have been reversed and are no longer active. If the search results are **NULL**, PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH searches for reversal events related to a suspended payment. If a reversal event is found, the suspended record is filtered and not returned.

How Suspended Payments Are Reversed

PCM_OP_BILL_REVERSE prepares information for PCM_OP_BILL_REVERSE_PAYMENT to perform reversals. PCM_OP_BILL_REVERSE reverses payments during the following payment suspense operations:

- A payment is recycled to or from suspense.
- A suspended payment is removed from BRM as unallocatable.

How Payments Are Reversed During Recycling

If PCM_OP_BILL_REVERSE is called from PCM_OP_PYMT_RECYCLE_PAYMENT, the reversal is due to recycling, and PCM_OP_BILL_REVERSE performs the following operations:

1. Assigns the reversal TRANS_ID value for each recycled payment.
2. Opens a transaction and checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled.
3. Calls PCM_OP_BILL_REVERSE_PAYMENT to perform the reversal and create the associated objects in the database. For each **/event/billing/reversal** object created, PIN_FLD_PAYMENT_TRANS_ID is set to the transaction ID of the recycled payment. This opcode populates the reversal flist with each recycled payment's TRANS_ID value.
4. Populates the payment batch with the sum of the reversal flist.
5. If the reversal was performed as part of recycling multiple distributed payments back into suspense, returns the recycled payments' reversal information in the PIN_FLD_MULTI_RESULTS array.

How Payments Are Removed As Unallocatable

PCM_OP_BILL_REVERSE performs the following operations to remove payments from BRM as unallocatable:

1. Assigns the reversal event a TRANS_ID value.

2. Opens a transaction and checks the value of the **PaymentSuspense** business parameter to determine whether Payment Suspense Manager is enabled.
3. Checks to see whether the following criteria are met:
   - The payment is a suspended payment. To do so, it compares the account POID in the flist with the account POID of the payment suspense account in the **/config/psm** object.
   - The payment is active.
   - The value of the PIN_FLD_FLAGS field is set to PIN_REVERSE_FLAG_REVERSE_AS_UNALLOCATED (**1**).
4. Calls PCM_OP_BILL_REVERSE_PAYMENT to reverse the payment in the payment suspense account and create the associated objects in the database. For the **/event/billing/reversal** object, PIN_FLD_PAYMENT_TRANS_ID is set to the transaction ID of the suspended payment. This opcode populates the reversal flist with the payment's TRANS_ID value.
5. Populates the payment batch with the sum of the reversal flist.
6. Validates the reverse payment operation and creates the reversal batch event in the database.

The result of the reversal is returned in the PIN_FLD_RESULTS field of PCM_OP_BILL_REVERSE. The reversal will not be allowed if either of the following conditions is true:

- The payment is not a suspended payment.
- The payment is not an active payment.

How BRM Handles Mandate Information for SEPA Processing

The following sections describe how BRM handles mandate information.

How BRM Registers a Mandate

To register the mandate information provided during customer account creation, BRM uses PCM_OP_CUST_SET_PAYINFO.

PCM_OP_CUST_SET_PAYINFO takes as input the mandate information such as the customer's IBAN and BIC and the creditor identification code.

This opcode adds the mandate by:

1. Creating a unique mandate reference number (UMR) for the mandate, if it is not provided in the input flist
2. Setting the status of the mandate to active
3. Creating the **/payinfo/sepa** object with the mandate information

PCM_OP_CUST_SET_PAYINFO calls PCM_OP_CUST_CREATE_PAYINFO, which calls PCM_OP_CUST_POL_VALID_PAYINFO, which validates that the format of the IBAN and BIC comply with the formats described in the SEPA rulebooks. Additional custom validations can be performed on the mandate information, such as country-specific validations.

How BRM Updates a Mandate

To update the mandate information, BRM uses the following opcodes:

- PCM_OP_CUST_AMEND_MANDATE, to update the customer information
- PCM_OP_CUST_AMEND_CREDITOR_INFO, to update the creditor information

PCM_OP_CUST_AMEND_MANDATE takes as input a reference to the **/payinfo/sepa** object and the mandate information to update.

This opcode updates the mandate by:

1. Updating the **/payinfo/sepa** object with the new mandate information
2. Updating the PIN_FLD_MANDATE_AMENDED field in the **/payinfo/sepa** object by using flags to indicate the fields that are amended
3. Generating the **/event/activity/sepa/mandate_amendment** event to record the mandate update

PCM_OP_CUST_AMEND_CREDITOR_INFO takes as input the creditor ID to be updated and the new creditor ID and creditor name.

This opcode updates the creditor information by:

1. Updating the **/config/creditor** object with the new creditor information
2. Determining if any **/payinfo/sepa** object exists that is associated with the original creditor ID and updating the **/payinfo/sepa** object with the new creditor information
3. In each **/payinfo/sepa** object that is updated, updating the PIN_FLD_MANDATE_AMENDED field by using flags to indicate the fields that are amended
4. Generating the **/event/activity/sepa/mandate_amendment** event to record the update

Updating Creditor Information

To update the creditor information, use the PCM_OP_CUST_AMEND_CREDITOR_INFO opcode. This opcode updates the creditor name and the creditor ID only.

PCM_OP_CUST_AMEND_CREDITOR_INFO uses the creditor ID to update the **/config/creditor** object with the new creditor information.

Any update to a creditor configuration triggers automatic updates to all the mandates with this creditor ID in the BRM database. For example, if you have multiple customers that have mandates with the same creditor ID, BRM automatically locates the customers' payment information and updates their mandates with the new creditor information.

How BRM Cancels a Mandate

To cancel a mandate, BRM uses PCM_OP_CUST_CANCEL_MANDATE.

PCM_OP_CUST_CANCEL_MANDATE takes as input the reference to the **/payinfo/sepa** object that contains the mandate to cancel.

This opcode cancels the mandate by:

1. Setting the PIN_FLD_MANDATE_STATUS field in the **/payinfo/sepa** object to PIN_MANDATE_STATUS_CANCELED
2. Setting the PIN_FLD_MANDATE_END_T field to the current time to record the time of cancellation

3. Calling PCM_OP_CUST_DELETE_PAYINFO to delete the **/payinfo/sepa** object. The opcode does not cancel the **/payinfo/sepa** object if it determines that it is associated with a bill unit or if a SEPA payment request is pending

22

Subscription Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) subscription opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Pricing Component Object Names
- Managing Charge Offers
- Managing Discount Offers
- Managing Bundles
- Managing Packages
- Managing Contracts
- Managing Purchase, Cycle, and Usage Charges in Purchased Offers
- Applying Recurring Charges
- Applying Folds
- Getting Data about Bundles, Charge Offers, Discount Offers, and Services
- Getting Information about Price Tags
- Applying Promotions for Special Dates, Events, or Actions
- Managing Promotions with Siebel CRM
- Managing Provisioning
- Validating Product Specification Attributes for Pricing Components

Opcodes Described in This Chapter

Table 22-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 22-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_ACT_SCHEDULE_CREATE	How BRM Transitions Accounts from Source Packages to Target Packages
PCM_OP_ACT_USAGE	Applying Cycle Forward Fees Applying Cycle Arrears Fees
PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS	Getting a Life-Cycle State
PCM_OP_BAL_GET_BALANCES	Getting a Life-Cycle State
PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	Getting a Life-Cycle State
PCM_OP_BILL_CYCLE_ARREARS	How Bundles Are Transitioned
PCM_OP_BILL_CYCLE_FORWARD	How Bundles Are Transitioned
PCM_OP_BILL_MAKE_BILL	Customizing How Folds Are Applied
PCM_OP_CONTRACT_CANCEL_CONTRACT	Canceling Contracts
PCM_OP_CONTRACT_CREATE_CONTRACT	Creating Contracts
PCM_OP_CONTRACT_MODIFY_CONTRACT	Modifying Contracts
PCM_OP_CONTRACT_POL_CANCEL_CONTRACT	Canceling Contracts
PCM_OP_CONTRACT_POL_POST_CREATE_CONTRACT	Creating Contracts
PCM_OP_CONTRACT_POL_PREP_CONTRACT	Creating Contracts Extending Subscriber Contracts
PCM_OP_CONTRACT_POL_VALID_CONTRACT	Creating Contracts
PCM_OP_CONTRACT_RENEW_CONTRACT	Renewing Contracts
PCM_OP_CUST_COMMIT_CUSTOMER	How Bundles Are Purchased Overriding Bundle Proration Settings at Customer Level Overriding Add-On Product Validity Dates in a Deal
PCM_OP_CUST_CREATE_CUSTOMER	About Optional and Required Service Types
PCM_OP_CUST_CREATE_PROFILE	Managing Provisioning
PCM_OP_CUST_DELETE_PROFILE	Managing Provisioning
PCM_OP_CUST_MODIFY_CUSTOMER	How Bundles Are Purchased Validating Changes to Bundles How BRM Transitions Accounts from Source Packages to Target Packages Overriding Add-On Product Validity Dates in a Deal
PCM_OP_CUST_MODIFY_PROFILE	Managing Provisioning
PCM_OP_CUST_POL_GET_DEALS	Getting Packages, Bundles, and Charge Offers for Purchase
PCM_OP_CUST_POL_GET_PLANS	Getting Packages, Bundles, and Charge Offers for Purchase
PCM_OP_CUST_POL_GET_PRODUCTS	Getting Packages, Bundles, and Charge Offers for Purchase
PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS	Getting a List of Packages and Bundles That an Account Owns
PCM_OP_CUST_POL_READ_PLAN	Getting Packages, Bundles, and Charge Offers for Purchase
PCM_OP_CUST_POL_TRANSITION_DEALS	Customizing Bundle Transitions Customizing Package Transitions without Configuring / transition Objects

Table 22-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_CUST_SET_STATUS	About Closing a Required Service How Bundle Dependencies Are Validated Validating Bundle Transitions
PCM_OP_CUST_UPDATE_SERVICES	About Closing a Required Service How BRM Transitions Accounts from Source Packages to Target Packages Getting a Life-Cycle State
PCM_OP_PROV_POL_UPDATE_SVC_ORDER	Managing GSM Service Provisioning
PCM_OP_PROV_PUBLISH_SVC_ORDER	Managing GSM Service Provisioning
PCM_OP_PROV_UPDATE_SVC_ORDER	Managing GSM Service Provisioning
PCM_OP_SUBSCRIPTION_CANCEL_DEAL	How Charge Offers Are Canceled How Bundles Are Purchased Validating Changes to Bundles How Bundles Are Modified How Bundles Are Canceled
PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT	How Discount Offers Are Canceled How Bundles Are Canceled
PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT	How Charge Offers Are Canceled Customizing Charge Offer Cancellation How Bundles Are Canceled Customizing How Folds Are Applied Getting a List of Provisioning Tags Customizing Provisioning When Canceling a Charge Offer
PCM_OP_SUBSCRIPTION_CHANGE_DEAL	How Bundles Are Modified
PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS	Validating Changes to Bundles
PCM_OP_SUBSCRIPTION_CRUD_OFFER_OVERRIDE	Overriding Charges and Discounts for a Period of Time
PCM_OP_SUBSCRIPTION_CYCLE_ARREARS	Applying Cycle Arrears Fees Calculating the Cycle Arrears Fee Customizing Which Balances to Fold When Charge Offers Are Canceled
PCM_OP_SUBSCRIPTION_CYCLE_FOLD	Applying Folds Customizing the Order to Apply Folds Customizing Which Balances to Fold When Charge Offers Are Canceled
PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Applying Cycle Forward Fees Calculating the Cycle Forward Fee How Cycle Fees Are Calculated for Backdated Cancellation or Inactivation Customizing Which Balances to Fold When Charge Offers Are Canceled
PCM_OP_SUBSCRIPTION_GET_HISTORY	Finding Events Associated with Bundles, Charge Offers, Discount Offers, and Services
PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS	Getting Information about Price Tags

Table 22-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS	Getting Purchased Offers Reading Data for All Valid Purchased Charge Offers and Discount Offers
PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT	Applying Promotions for Special Dates, Events, or Actions
PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING	How Charge Offers Are Canceled Customizing Provisioning When Canceling a Charge Offer
PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS	Getting a List of Provisioning Tags
PCM_OP_SUBSCRIPTION_POL_PRE_FOLD	Customizing How to Apply Folds Customizing How Folds Are Applied
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL	How Bundles Are Transitioned
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN	Customizing Bundle Transitions How BRM Transitions Accounts from Source Packages to Target Packages Customizing Package Transitions without Configuring / transition Objects About Providing Transition Rules Using Policy Opcodes
PCM_OP_SUBSCRIPTION_POL_PREP_FOLD	Customizing How to Apply Folds Customizing Which Balances to Fold When Charge Offers Are Canceled
PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING	Purchasing Charge Offers
PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING	Customizing Provisioning When a Charge Offer Is Purchased
PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT	Customizing Snowball Discounts
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL	Customizing Charge Offer Cancellation How Charge Offers Are Canceled
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT	Customizing Discount Offer Cancellation
PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD	Customizing How to Apply Folds Applying Folds Customizing the Order to Apply Folds
PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING	Purchasing Charge Offers How Discount Offers Are Purchased How Bundles Are Purchased Validating Product Specification Attributes for Pricing Components
PCM_OP_SUBSCRIPTION_PROVISION_ERA	Managing Provisioning

Table 22-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_PURCHASE_DEAL	Purchasing Charge Offers How Discount Offers Are Purchased How Bundles Are Purchased Validating Changes to Bundles How Bundles Are Modified Managing Purchase, Cycle, and Usage Charges in Purchased Offers Overriding Bundle Proration Settings at Customer Level Overriding Add-On Product Validity Dates in a Deal
PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT	How Discount Offers Are Purchased Managing Purchase, Cycle, and Usage Charges in Purchased Offers
PCM_OP_SUBSCRIPTION_PURCHASE_FEES	Applying Deferred Charge Offer Purchase Fees
PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT	Purchasing Charge Offers Backdating Charge Offer, Discount Offer, Bundle, or Package Purchases About Delayed Cycle Start and End Times Applying Purchase and Cycle Fees to the Balance How Bundles Are Purchased Managing Purchase, Cycle, and Usage Charges in Purchased Offers Customizing Provisioning When a Charge Offer Is Purchased
PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS	Getting a List of Bundles, Charge Offers, Discount Offers, and Services
PCM_OP_SUBSCRIPTION_SET_BUNDLE	Managing Promotions with Siebel CRM
PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS	How BRM Changes Discount Offer Status How Discount Offer Purchase, Cycle, and Usage Validity Is Modified
PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO	How Discount Offer Purchase, Cycle, and Usage Validity Is Modified How BRM Changes Discount Offer Status About Backdated Purchase, Usage, or Cycle End Dates
PCM_OP_SUBSCRIPTION_SET_PRODINFO	How Charge Offers Are Modified How BRM Changes Charge Offer Status Changing the Purchase, Usage, and Cycle Start and End Times Calculating the Cycle Forward Fee Calculating the Cycle Arrears Fee About Backdated Purchase, Usage, or Cycle End Dates
PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS	How BRM Changes Charge Offer Status Changing the Purchase, Usage, and Cycle Start and End Times
PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION	How Bundles Are Transitioned

Table 22-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_SUBSCRIPTION_TRANSITION_DEAL	Purchasing Charge Offers Applying Purchase and Cycle Fees to the Balance How Charge Offers Are Canceled How Bundles Are Transitioned Customizing Bundle Transitions
PCM_OP_SUBSCRIPTION_TRANSITION_PLAN	How BRM Transitions Accounts from Source Packages to Target Packages About Providing Transition Rules Using Policy Opcodes Customizing Account Bundles during Package Transitions
PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY	How Bundles Are Purchased Validating Changes to Bundles How Bundle Dependencies Are Validated Validating Bundle Transitions
PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY	Validating Discount Offer Dependencies

Pricing Component Object Names

When you publish pricing components to the BRM database, the pricing components are stored as objects in the BRM database. Table 22-2 shows the pricing component object names. Developers and database managers must know these object names.

Table 22-2 Pricing Component Object Names

Pricing Component	Object
bundle	**/deal** Bundles owned by a customer are stored as **/purchased_deal** objects.
charge	**/rate_plan**
charge pricing	**/rate**
charge offer	**/product** Charge offers owned by a customer are stored as **/purchased_product** objects.
charge selector	**/rate_plan_selector**
discount offer	**/discount** Discount offers owned by a customer are stored as **/purchased_discount** objects.
product specification attribute	**/offer_attribute_group** Attribute names and values are stored as **/offer_attribute_group** arrays. Pricing components contain a PIN_FLD_ATTRIBUTE_OBJ field to point to these arrays.
package	**/plan**
package list	**/group/plan_list**

Managing Charge Offers

See the following topics:

- Purchasing Charge Offers
- Backdating Charge Offer, Discount Offer, Bundle, or Package Purchases
- Applying Purchase and Cycle Fees to the Balance
- How Charge Offers Are Modified
- How BRM Changes Charge Offer Status
- How Charge Offers Are Canceled
- Customizing Charge Offer Cancellation
- Customizing How to Apply Folds

Purchasing Charge Offers

To purchase a charge offer, use PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT.

> **Note:**
>
> Do not call PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT directly. This opcode is always called by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT is called to purchase charge offers for the account or service object specified in the input flist.

> **Note:**
>
> - If a **/service** object is specified, the charge offer is purchased for that service and is a *service charge offer*. The **/service** object must belong to the account.
> - If the **/service** object is NULL, the charge offer is purchased for the **/account** object and is an *account bundle*.

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT performs these operations:

1. If billing is due, triggers billing.
2. Opens a transaction.
3. Checks the value of the PIN_FLD_MODE field.
   - If the value is **0**, continues with the purchase.
   - If the value is **1**, **2**, or **3**, processes the purchase as described in "Purchasing the Same Charge Offer Multiple Times".
4. Validates any product specification attributes on the charge offer by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *BRM Setting Up Pipeline Pricing*.

5. Retrieves the charge offer's package ID, name, and type, which are passed in the input flist, and opens a bundle instance. For information about how a bundle is generated, see PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.
6. Validates that the charge offer is available for purchase. A charge offer is *not* available for purchase when any of the following apply:
   - The charge offer's validity period has not yet started (the PURCHASE_START_T value is greater than the current time).
   - The charge offer's validity period has already ended (the PURCHASE_END_T value is less than or equal to the current time).
   - The cycle or usage start time is less than the purchase start time.
   - The cycle or usage end time is greater than the purchase end time.
   - There is a PIN_FLD_PERMITTED field for the charge offer but the POID type is not explicitly permitted.
   - The purchase quantity is **0** or is not passed.
   - The purchase quantity is less than the minimum purchase quantity set in your product offerings.
   - The purchase quantity is greater than the maximum purchase quantity set in your product offerings.
   - An attempt is made to purchase a partial quantity.
   - The total quantity purchased for this charge offer exceeds the maximum ownership quantity in your product offerings for this account or service. This applies only to subscription charge offers that have a maximum ownership quantity specified.
   - The total quantity purchased for this charge offer is less than the minimum ownership quantity in your product offerings for this account or service. This applies only to subscription charge offers that have a minimum ownership quantity specified.
7. If the charge offer purchase is backdated, validates that:
   - The date to which the charge offer purchase is backdated is not prior to the G/L posting date.
   - The date to which the charge offer purchase is backdated is not prior to the effective date of the account or service.
   - The date to which the charge offer purchase is backdated is not prior to the date of the last status change of the account or service.
8. If the **purchase product** provisioning tag is set for the charge offer, calls PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING. See "Customizing Provisioning When a Charge Offer Is Purchased".
   If a **/config/provisioning_tag** object is associated with the charge offer, PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING runs the opcodes specified in that object to run at charge offer purchase time. The opcode always runs the opcodes specified by the __DEFAULT__ provisioning tag.
9. Determines the various start and end dates for the charge offer. See "Handling Purchase, Cycle, and Usage Start and End Times".
10. Verifies the charge offer purchase with the specified account.
11. Applies the charge offer to the account:
    - If it is a subscription charge offer, generates a **/purchased_product** object with a pointer to the **/account** object.

- If it is an item charge offer, creates an audit record for the **/au_purchased_product** object.

> **Note:**
>
> Item charge offers cannot be deferred. The audit object is used for rerating events.

12. Sets the purchase, cycle, and usage discounts, if applicable.
13. Sets the charge offer status in **/purchased_product**.
14. Checks the PIN_FLD_FLAGS value set by PCM_OP_SUBSCRIPTION_TRANSITION_DEAL to determine whether the charge offer purchase is due to a bundle or package transition. If so, sets the value to PIN_TRANS_WAIVE_PURCHASE_FEES.
15. Applies the appropriate purchase and cycle fees to the balance.
16. Updates the charge offer flags for applying purchase and cycle fees.
17. Creates the audit log event object (**/event/billing/product/action/purchase**) after the charge offer is added to the account and all applicable fees are applied.
18. Closes the transaction.

If the charge offer purchase is successful, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT returns a confirmation message.

During a bundle or package transition, additional validations are performed that may prevent a successful purchase:

- If the service contains at least one required bundle, the service flag is set to **Required**. If all bundles for the service are optional, the service flag is set to **Optional**. A required service cannot be canceled; therefore, the purchase portion of the transition does not occur.
- If the system flag **validate_deal_dependencies** is **1**, BRM checks to see whether prerequisites exist or whether the bundles are mutually exclusive before PCM_OP_SUBSCRIPTION_PURCHASE_DEAL is called.
- If the permitted field in the package is **service_type**, the flag is set to **PBS**.

Applying Deferred Charge Offer Purchase Fees

To apply deferred purchase fees, use PCM_OP_SUBSCRIPTION_PURCHASE_FEES.

By default, purchase fees are applied at the time of charge offer purchase. However, you can defer the purchase fees to a later date. For example, a customer can sign up for a charge offer that is not available until a later date. The charge offer's purchase fees are deferred and applied when the charge offer becomes available.

PCM_OP_SUBSCRIPTION_PURCHASE_FEES is called by the **pin_cycle_fees** utility to apply deferred purchase fees. When **pin_cycle_fees** is run with the **-purchase** parameter, it searches for **/account** objects with an expired PURCHASE_START_TIME value. For each account that it finds, it calls PCM_OP_SUBSCRIPTION_PURCHASE_FEES to apply the purchase fees.

PCM_OP_SUBSCRIPTION_PURCHASE_FEES applies deferred purchase fees only for charge offers with an expired purchase start time.

> **Note:**
>
> For item charge offers, the **/purchased_product** object is removed after the deferred fees are applied.

After the purchase fees are applied to the account, an **/event/billing/purchase_fee** object is created for auditing purposes.

If the purchase fee is applied successfully, PCM_OP_SUBSCRIPTION_PURCHASE_FEES returns the POIDs of the **/account** object and the **/event/billing/purchase_fee** object.

About Delayed Cycle Start and End Times

If you configure delayed purchase, cycle, or usage start and end times when you set up your price list, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT sets the delayed start and end times in the **/purchased_product object** when the charge offer is purchased.

The day of month to which the start date is set depends on whether you configure to align the start and end dates with the accounting cycle by setting the **CycleDelayAlign** business parameter to **enabled**.

For example, you configure a charge offer to start relative to the purchase date, and you set the relative period in the pricing package to one accounting cycle. If the account creation day is January 5, the billing day of month is 5, and the bundle is purchased on January 20, then the start date is set as follows:

- If start and end dates are not aligned with the accounting cycle, the start date is set to February 20.
- If start and end dates are aligned with the accounting cycle, the start date is set to February 5 if the accounting cycle is short or March 5 if the accounting cycle is long.

Handling Purchase, Cycle, and Usage Start and End Times

When using PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT, you pass the purchase, cycle, and usage start and end times. You can specify fixed dates, start on first usage and end relative to the start date, or start relative to the purchase date and end relative to the start date.

The start and end times are passed in by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. The start and end times are specified in the same way for PCM_OP_SUBSCRIPTION_PURCHASE_DEAL and PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT.

For information on setting purchase, cycle, and usage start and end times, see "Managing Purchase, Cycle, and Usage Charges in Purchased Offers".

If the system is configured for time-stamp rounding, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT rounds the start and end times to 00:00:00 hours.

However, when a charge offer's purchase, cycle, and usage start and end units are set to 1 (seconds), 2 (minutes), or 3 (hours), and the validity period is less than 24 hours, time stamps are not rounded, regardless of your system configuration. If the validity is greater than 24 hours, or crosses the threshold of a calendar day, the purchase, cycle, and usage end time stamps are rounded when calculating the scale to determine the cycle fee amount to charge. For example, at 23:00:00 on June 10, a customer purchases a charge offer that is valid for

three hours. Because the validity crosses into the next day, the timestamps for the scale calculation are rounded to 00:00:00 on June 11 rather than 02:00:00.

Purchasing the Same Charge Offer Multiple Times

Customers can purchase the same charge offer multiple times using PCM_OP_SUBSCRIPTION_PURCHASE_DEAL, which accepts PIN_FLD_MODE as part of the input flist and passes it to PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT.

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT checks the value of PIN_FLD_MODE to determine how to process the purchase.

If the value is **0**, the normal product purchase flow proceeds, purchasing the charge offer as a new, unrelated purchased product. This is the default option.

If the value is **1**, **2**, or **3**, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT uses the account, service, and bundle in the input flist to search for an existing purchased product. If no matching product is found, the product is treated as a new subscription, and the purchase flow continues as normal.

If there is a matching product and the value of PIN_FLD_MODE is **1** or **2**:

> **Note:**
>
> This is applicable only to one-time offers and not recurring offers.

1. PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT determines whether the additional charge offer has been purchased within the grace period and then does one of the following:
   - If the repurchase occurs within the defined grace period, the existing charge offer is extended and the opcode proceeds to the next step.
   - If the repurchase occurs after the defined grace period, the additional charge offer is treated as a new, unrelated purchased product.
2. PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT calculates the new balance validity and purchase, usage, and cycle end dates as follows:
   - If PIN_FLD_MODE is set to **1**, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT compares the existing validity period to the new one, and chooses the later of the two.

     For example, an existing balance is valid until June 7th. The new charge offer is purchased on June 3rd, and is valid for 7 days. The end of the new validity period is later than the existing one, so the new end date of June 10th will be used.
   - If PIN_FLD_MODE set to **2**, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT adds the new validity period to the existing one.

     For example, an existing balance is valid until June 7th. The new charge offer is purchased on June 3rd, and is valid for 7 days. The 7 days are added to the end of the existing validity period, so the new end date of June 14th will be used.
3. PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT calls PCM_OP_RATE_EVENT to add the balance impacts from the new charge offer as sub-balances of the existing balance, and sets the validity of the total balance calculated according to the different PIN_FLD_MODE values.
4. PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT then calls PCM_OP_SUBSCRIPTION_SET_PRODINFO to update the purchase, usage, and cycle

end dates of the product to the new end dates calculated according to the different PIN_FLD_MODE values.

If there is a matching product and the value of PIN_FLD_MODE is **3**:

1. PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT calls PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT to cancel the existing product and prorate any remaining balances.
2. The normal product purchase flow continues, purchasing the charge offer as a new, unrelated purchased product.

Backdating Charge Offer, Discount Offer, Bundle, or Package Purchases

To backdate a charge offer, discount offer, bundle, or package purchase, enter the backdated time in the PIN_FLD_ END_T field of the corresponding opcode. For example, to backdate a charge offer purchase, enter the date to which you want to backdate the purchase in the PIN_FLD_ END_T field of the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode.

When you backdate a charge offer, discount offer, bundle, or package purchase, BRM does the following:

- Sets the purchase, usage, and cycle start dates to the backdated date unless they are explicitly set to a different date.
- Prorates the cycle fee based on the proration settings.
- Applies the purchase fee on the backdate date.

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

- The backdated purchase date is prior to the G/L posting date.
- The backdated purchase date is prior to the account or service's effective date.
- The backdated purchase date is prior to the date of the last status change of the account or service.

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

> **Note:**
>
> – BRM does not apply a fold, rollover charge offer, or billing time discount for the past accounting cycles when you backdate the purchase of the fold, rollover charge offer, or billing time discount.
>
> – Backdating discount offers with discount offer proration options is supported only for discount offers that are prorated.

How Sub-balance Buckets Are Created for Backdated Charge Offer Purchases

When a backdated charge offer purchase spans beyond the current accounting cycle, multiple cycle events and sub-balance buckets are created. That is, if a backdated purchase of a

charge offer spans multiple cycles, one cycle event is created for each cycle and the free sub-balance buckets are split into multiple cycles.

For example, consider that you create an account on January 1. On April 1, you backdate the purchase of a charge offer to January 15. The charge offer grants 3600 Anytime Minutes monthly. BRM creates the following three sub-balances for the three cycles:

- 1800 Anytime Minutes for January 15 to February 1.
- 3600 Anytime Minutes for February 1 to March 1.
- 3600 Anytime Minutes for March 1 to April 1.

> **Note:**
>
> If you backdate the purchase of a discount offer on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be applied only from the current accounting cycle.

How Cycle Fees Are Calculated for Backdated Purchase or Activation

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

When you backdate a purchase or activation, the opcodes determine the scale values that are used for prorating the amount to be charged. The charges for the events that occurred during the backdated period are prorated and applied accordingly. These charges for the backdated period appear as normal recurring fees in the bill of the current cycle.

When you backdate the purchase of a charge offer that includes cycle forward events, cycle forward fees are prorated and applied from the backdated purchase date through the current date.

Applying Purchase and Cycle Fees to the Balance

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT handles the purchase and cycle fees for the charge offer as follows:

- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT does not apply either deferred purchase or cycle forward fees. These fees are applied when the **pin_cycle_fees** utility is run as part of the **pin_bill_day** billing script. See "Applying Deferred Charge Offer Purchase Fees".
- If the charge offer has a purchase fee and if the purchase start time is not deferred, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies the purchase fee and creates the associated **/event/billing/product/fee/purchase** object.

> **Note:**
>
> Unlike flexible cycle forward events, BRM does not support custom events for purchase fee events. Purchase fees associated with a charge offer are stored in **/event/billing/product/fee/purchase** object.

- If the charge offer has a cycle forward fee and if the cycle start time is not deferred, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies the cycle forward fee by calling the necessary opcodes. This creates the **/event/billing/product/fee/cycle/cycle_forward**_feetype object, where feetype is the type of cycle forward fee.
- If the charge offer is purchased as part of a bundle transition, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT can waive the purchase fees based on the PIN_FLD_FLAGS value passed in by PCM_OP_SUBSCRIPTION_TRANSITION_DEAL. For more information, see "How Bundles Are Transitioned".

How Charge Offers Are Modified

To modify a charge offer owned by a customer, use PCM_OP_SUBSCRIPTION_SET_PRODINFO.

PCM_OP_SUBSCRIPTION_SET_PRODINFO performs these operations:

1. If billing is due, triggers billing.
2. Opens a transaction.
3. For backdating operations, validates and sets the charge offer's purchase, cycle, or usage start and end dates to a backdated date. See "About Backdated Purchase, Usage, or Cycle End Dates".
4. Modifies the charge offer information as specified. For information on how PCM_OP_SUBSCRIPTION_SET_PRODINFO handles start and end dates, see "Changing the Purchase, Usage, and Cycle Start and End Times".
5. Creates an audit record object (**/au_purchased_product**) used for rerating events.
6. Applies the charge offer changes to the charge offer object (**/purchased_product**) owned by the account.
7. Calculates and applies appropriate cycle fees for the event to the balance and, if necessary, calls other opcodes to record the cycle events. See "Calculating the Cycle Forward Fee".
8. Creates the **/event/billing/product/action/modify** event objects.
9. Closes the transaction.

If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_SUBSCRIPTION_SET_PRODINFO returns the entire event flist for the events created as a result of the modification. Otherwise, it returns the event POID of all event objects created as a result of the modification.

An error occurs in the following cases:

- If an attempt is made to change the cycle start date to a date earlier than any period for which cycle forward or arrears fees have already been applied.
- If an attempt is made to change the cycle end date to a date before the current time but after the end of any period for which cycle forward or cycle arrears fees have already been applied.

How BRM Changes Charge Offer Status

To set the status of a **/purchased_product** object owned by an account, use PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS.

PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is called in the following cases:

- When the status of an account or service is changed.
- When a charge offer status is changed. You might need to change only the charge offer status itself if, for example, the charge offer was purchased as inactive because of future provisioning and you activate it later, or if a customer's subscription was suspended, and is now being reactivated.

PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS performs the following tasks:

1. Opens a transaction.
2. Retrieves the new status from the PIN_FLD_STATUSES array in the input flist. If the charge offer status change is due to an account or service status change, PIN_STATUS_FLAG_DUE_TO_ACCOUNT is passed in the PIN_FLD_STATUS_FLAGS field of the array.
3. Reads the old status of the charge offer from the database.
4. If PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is *not* called in CALC_ONLY mode, applies the new status to the charge offer.
5. If the charge offer is inactive due to future provisioning and is now being activated, then PCM_OP_SUBSCRIPTION_SET_PRODINFO is called to modify the purchase start date and time to the date and time of reactivation. If the original cycle and usage start date is earlier than the new purchase start date, PCM_OP_SUBSCRIPTION_SET_PRODINFO also resets the cycle and usage start date and time to the new purchase start date.
   If PCM_OP_SUBSCRIPTION_SET_PRODINFO is not called, an **/au_purchased_products** audit object is generated, which is used for rerating events.
6. If the charge offer is inactive due to a suspended subscription that is now being reactivated and PIN_FLD_RENEWAL_MODE and PIN_FLD_CYCLE_FEE_FLAGS are both set to **1**, then PCM_OP_SUBSCRIPTION_SET_PRODINFO is called to modify the cycle end date and time to align with the date and time of the reactivation.
   If the PIN_FLD_CALENDAR_DOM field is also set, the end date and time will be whichever is later, the value of PIN_FLD_CALENDAR_DOM or the reactivation date.
7. If PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is called while activating an account, sets the purchase time to NOW and sets the usage and cycle fees by calling PCM_OP_SUBSCRIPTION_SET_PRODINFO.
8. If the charge offer status change is backdated, validates that:
   - The date to which the charge offer status change is backdated is not prior to the G/L posting date.
   - The date to which the charge offer status change is backdated is not prior to the account or service's effective date.
   - The date to which the charge offer status is backdated is not prior to the last status change of the account or service.
9. Creates an audit log event object (**/event/billing/product/action/modify/status**).
10. Closes the transaction.

If the PCM_OP_CALC_ONLY flag is set when calling PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS, it returns the entire event flist for the events created as a result of the modification. Otherwise, it returns the event POIDs of all event objects created as a result of the modification.

How Charge Offers Are Canceled

To cancel a charge offer, use PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT. This opcode cancels the charge offers associated with the **/account** object specified in the input flist.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT is called by:

- PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel each charge offer associated with a specific bundle. Depending on how many charge offers are included in the bundle, PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT may be called more than once.
- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT to cancel an existing charge offer when the same one is purchased multiple times and the charge offer's PIN_FLD_MODE field is set to **3**.

The input flist identifies the charge offers to cancel in the PIN_FLD_OFFERING_OBJ (**/purchased_product** POID) field.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT performs the following tasks:

1. Closes billing if the account balance is affected.
2. Opens a transaction.
3. Verifies that the charge offer is valid for cancellation.
4. Checks if the corresponding bundle is a required bundle. If so, does not perform the cancellation.
5. Calls PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL to determine whether to cancel and delete the charge offer, to cancel without deleting the charge offer, or to not allow the cancellation. If the corresponding bundle has any dependencies, the charge offer cannot be deleted.

   > **Note:**
   >
   > By default, BRM retains canceled charge offers. You can change this behavior by customizing PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL. See "Customizing Charge Offer Cancellation".

6. If the cancel product provisioning tag is set for the charge offer, PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT calls PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING to clear fields in the service object. See "Customizing Provisioning When Canceling a Charge Offer".
   If a **/config/provisioning_tag** object is associated with the charge offer, PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT runs the opcodes specified in that object to run at charge offer cancellation time. The opcode always runs the opcodes specified by the __DEFAULT__ provisioning tag.
7. Validates the charge offer quantity requested for cancellation. The quantity must be less than or equal to the quantity owned by the account.

> **Note:**
>
> When the quantity to cancel is less than the quantity owned, the charge offer quantity is modified; the charge offer is not canceled.

8. If the charge offer cancellation is backdated, validates that:
   - The date to which the charge offer cancellation is backdated is not prior to the G/L posting date·
   - The date to which the charge offer cancellation is backdated is not prior to the effective date of the account or service.
   - The date to which the charge offer cancellation is backdated is not prior to the last status change date of the account or service.
9. Sets the charge offer's purchase, usage, and cycle end dates to the cancellation date.
10. Reads the PIN_FLD_FLAGS value passed in from PCM_OP_SUBSCRIPTION_TRANSITION_DEAL to determine if cancellation fees should be waived.
11. Applies the following fees, if applicable:
    - If the charge offer is active and has any cycle forward fee associated with it, and if the charge offer is canceled in the middle of the cycle, the cycle forward fee is refunded for the unused portion of the cycle and an appropriate cycle forward event is generated.
    - If the charge offer is active and has a cycle arrears fee associated with it, and if the charge offer is canceled in the middle of the cycle, the cycle arrears fee is charged for the used portion of the cycle and an **/event/billing/product/fee/cycle/cycle_arrears** event object is generated.
    - If the charge offer has a cancellation fee, the cancellation fee is applied and an **/event/billing/product/fee/cancel** event object is generated.

    > **Note:**
    >
    > If the charge offer's validity period is configured to start on first usage and has not yet been set, the charge offer has not been activated. In this case, cycle fees are not charged.
12. Refunds the charge offer's cycle fees, and then reapplies them based on the new validity.
13. Creates the audit log **/event/billing/product/action/cancel** object to record the cancellation of the charge offer.
14. Returns the POIDs of all event objects created as a result of the modification.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT fails in the following cases:

- If the specified charge offer in the input flist does not exactly match the **/purchased_product** object.
- If the cancellation quantity is not passed in the input flist or is passed as **0**.
- If the charge offer pricing quantity is less than the cancellation quantity specified in the input flist.

Customizing Charge Offer Cancellation

To customize charge offer cancellation, use PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL.

This opcode is called by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT.

By default, when a charge offer is canceled, the **/purchased_product object** is not deleted, but its status is set to **Canceled**. In addition, the end date is set to the charge offer cancellation date. PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL allows you to specify whether the **/purchased_product** object is deleted or remains with a status of **Canceled**.

> **Note:**
>
> BRM requires information stored in the **/purchased_product** object to rate events when you use delayed billing and when events are rerated. In these cases, **/purchased_product** objects should not be deleted.

You can customize the behavior of PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL by setting the PIN_FLD_ACTION field to one of these values:

- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY, which sets the status of the **/purchased_product** object to **Canceled** and does not delete the **/purchased_product** object when canceled.
- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE, which deletes the **/purchased_product** object when it is canceled.
- PIN_BILL_CANCEL_PRODUCT_ACTION_DONOT_CANCEL, which stops the charge offer cancellation.

> **Note:**
>
> The **pin_bill.h** header file defines all action strings.

By default, PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL reads the provisioning tag of the **/purchased_product** object. If a provisioning tag is configured for the charge offer, the action is set to PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY. If there is no provisioning tag, the action is set to PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE.

Customizing How to Apply Folds

You can customize how folds are applied:

- Use PCM_OP_SUBSCRIPTION_POL_PRE_FOLD to apply custom logic to how folds are applied.
- Use PCM_OP_SUBSCRIPTION_POL_PREP_FOLD to customize the list of balance impacts that must be folded when a charge offer is canceled.

- Use PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD to customize the order in which folds are applied. By default, an account's balances are folded in ascending order based on the balance element ID.

Managing Discount Offers

See the following topics:

- How Discount Offers Are Purchased
- How Discount Offer Purchase, Cycle, and Usage Validity Is Modified
- How BRM Changes Discount Offer Status
- How Discount Offers Are Canceled
- Validating Discount Offer Dependencies
- Customizing Discount Offer Cancellation

How Discount Offers Are Purchased

To purchase a discount offer, use PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT.

> **Note:**
>
> Do not call PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT directly. This opcode is always called by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT performs these operations:

1. Closes the bill.
2. If billing is due, triggers billing.
3. Opens a transaction.
4. Reads the discount offer information.
5. Checks the value of the PIN_FLD_MODE field:
   - If the value is **0**, the purchase continues as normal.
   - If the value is **1**, **2**, or **3**, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT uses the account, service, and bundle in the input flist to search for an existing purchased discount.
     If no matching discount is found, the discount is treated as new, and the purchase continues as normal.

     If a matching discount is found and the value of PIN_FLD_MODE is **1** or **2**, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT determines whether the additional discount offer has been purchased within the grace period. If the repurchase occurs within the defined grace period, the existing discount offer is extended and the opcode proceeds to the next step. If the repurchase occurs after the defined grace period, the additional discount offer is treated as new, and the purchase continues as normal.

> **Note:**
>
> This is applicable only to one-time offers and not reccuring offers.

- If PIN_FLD_MODE is set to **1**, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT compares the existing validity period to the new one, and chooses the later of the two.

  For example, an existing balance is valid until June 7th. The new discount offer is purchased on June 3rd, and is valid for 7 days. The end of the new validity period is later than the existing one, so the new end date of June 10th will be used.

- If PIN_FLD_MODE is set to **2**, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT adds the new validity period to the existing one.

  For example, an existing balance is valid until June 7th. The new discount offer is purchased on June 3rd, and is valid for 7 days. The 7 days are added to the end of the existing validity period, so the new end date of June 14th will be used.

- If PIN_FLD_MODE is set to **3**, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT calls PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT to cancel the existing discount, then continues with the purchase as normal.

6. Performs the following validations:
   - Date validation, to make sure that the charge offer is available for purchase.
   - Status validation, to check whether the discount offer is active or inactive.
   - Ownership quantity validation, to make sure that the discount offer instance purchase quantity is within minimum and maximum limits for ownership.
   - Purchase quantity validations, to make sure that a partial quantity purchase is not made and to check that the purchased quantity is within the minimum and maximum limits for purchase.
   - product specification attribute validation, by calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *BRM Setting Up Pipeline Pricing*.
7. Calculates the start and end dates of the purchase/cycle/usage events based on either CSR customization or pricing component details. If passed relative dates, calculates fixed dates. If BRM is configured for time stamp rounding, all start and end dates round to 00:00:00 hours. For more information, see "Managing Purchase, Cycle, and Usage Charges in Purchased Offers".
8. If the discount offer purchase is backdated, validates that:
   - The date to which the discount offer purchase is backdated is not prior to the G/L posting date.
   - The date to which the discount offer purchase is backdated is not prior to the effective date of the account or service.
   - The date to which the discount offer purchase is backdated is not prior to the date of the last status change of the account or service.

> **Note:**
>
> If you backdate the purchase of a discount offer on a cycle forward or cycle arrears event to a previous accounting cycle, the discount is applied only from the current accounting cycle.

9. If a **/config/provisioning_tag** object is associated with the discount offer, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT runs the opcodes specified in that object to run at discount offer purchase time.
10. If applying discount offer validity rules, sets the cycle and usage start and end dates to one of the following:
    - The first day of the next accounting cycle (**no discount, valid from the middle of the cycle**).
    - The discount offer purchase date (**prorated discount, valid from the middle of the cycle**).
    - The first day of the current accounting cycle (**full discount, valid from the middle of the cycle**).
    - The purchase date of the discount offer (**Prorated discount, valid only part of the cycle**).
11. Determines whether the discount offer instance is to be created. If the opcode is not deferring item discounts or if the opcode has no associated purchase fee, creates a **/purchased_discount** object with a pointer to the **/account** object. For some item discounts, an instance is created, but it is removed after the deferred date when the purchase fee is applied.
12. If purchasing a discount offer in the middle of a cycle and the discount offer contains discounts for any cycle forward event types, refunds the discounted portion of the cycle forward fee as follows:
    a. Checks all the discounts to see whether they support any cycle forward monthly event types.
    b. For each cycle forward event type supported by a discount, checks the charges arrays of all **/purchased_product** objects for an array element of the cycle forward event type.
    c. Calculates the scale value for the portion of the cycle for which the discount is valid.
    d. Sends a request to rate normal prorated events, and calculates the resulting prorated balance impact.
    e. Sends the event with an inverted flag set for recording.
13. Generates an audit log for the discount offer purchase to record an **/event/billing/discount/action/purchase** event.
14. Prepares the return flist.
15. Closes the transaction.

If the discount offer purchase is successful, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT returns a message confirming the success.

How Discount Offer Purchase, Cycle, and Usage Validity Is Modified

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO is called to change the start or end time of an account's discount offer purchase, cycle, or usage period. This opcode is called by PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS when the status of an account's discount offer is changed: for example, when you set a discount offer's status to **Inactive** when you purchase it and activate it at a later time.

After a discount offer is purchased and the start and end times are set, you can change the start and end times to specify only other fixed dates.

To change the discount offer's purchase, cycle, or usage validity period, specify the new start and end dates in purchase, cycle, and usage START_T and END_T fields in the PIN_FLD_DISCOUNTS array in the input flist. The cycle and usage start times must be on or later than the purchase start time, and the cycle and usage end times must be on or earlier than the purchase end time.

> **Note:**
>
> Do not change a discount offer's purchase, cycle, or usage start time after the discount offer has been applied to the account; otherwise, the discount is applied incorrectly.

If you change the purchase, cycle, or usage start time from first usage to a fixed date, the opcode generates an event that causes ECE to update the validity period.

The start and end times are stored in the account's **/purchased_discount** object.

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO performs the following tasks:

1. Resets the **/purchased_discount** object's purchase, cycle, and usage start and end times to the times specified in the input flist.

   If discount offer validity rules are set, PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO sets the start and end times accordingly.
2. For backdating operations, validates and sets the discount offer's purchase, cycle, and usage end dates to a backdated date.
3. Applies cycle fees or refunds discounted cycle fee amounts.

   > **Note:**
   >
   > When only the purchase period is changed and the cycle period starts on first usage, if the validity of the cycle period has not been initialized by a first-usage event, PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO does not apply cycle fees or refund discounted cycle fee amounts.
4. Generates an **/event/billing/discount/action/modify** event to record the modification.

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO returns the POID of the **/event/billing/discount/action/modify** event.

How BRM Changes Discount Offer Status

To set the status of a **/purchased_discount** object owned by an account, use PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS.

This opcode is called when the status of a discount offer is changed. This can occur in the following cases:

- When the status of the account or service that owns the discount offer is changed. In this case, the discount offer's status is changed to the status of the account or service.
- When the status of a discount offer is changed to inactive to active.

PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS performs the following tasks:

1. Validates and sets the new status. The new status cannot be the same as the old status.
2. Calls PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO to reset the **/discount** object's purchase, cycle, and usage start and end dates.

   > **Note:**
   >
   > If the discount offer purchase, cycle, and usage start or end dates are already set to a later date, or the CM is configured to not reset dates, the dates are not reset.

3. If the status change of the discount offer is backdated, PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS validates that:
   - The date to which the discount offer status change is backdated is not prior to the general ledger (G/L) posting date.
   - The date to which the discount offer status change is backdated is not prior to the account or service effective date.
   - The date to which the discount offer status is backdated is not prior to the last status change of the account or service.
4. Generates an **/event/billing/discount/action/modify/status** event to record the status change.

If successful, PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS returns the POID of the **/event/billing/discount/action/modify/status** event.

How Discount Offers Are Canceled

To cancel a discount offer instance, use PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT.

PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT is called by:

- PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel each discount offer associated with a specific bundle. Depending on how many discount offers are included in the bundle, PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT may be called more than once.
- PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT to cancel discount offers when they are purchased subsequent times as replacements. This happens when the PIN_FLD_MODE field for the discount offer is set to **3**.

This opcode cancels the discount offers for the **/account** object or **/service** object specified in the input flist.

> **Note:**
>
> If you specify a NULL **/service** object, you cancel discount offers associated with the **/account** object. Otherwise, you cancel discount offers associated with the **/service** object.

You identify discount offers to cancel in the PIN_FLD_OFFERING_OBJ (**/purchased_discount** POID) field in the input flist.

To cancel a discount offer instance, PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT performs the following tasks:

1. Validates the discount offer quantity requested for cancellation. The quantity must be less than or equal to the quantity owned by the account or service.

   > **Note:**
   >
   > When the quantity to cancel is less than the quantity owned, the discount offer quantity in the instance is decreased; the discount offer is not canceled.

2. If the discount offer cancellation is backdated, validates that:
   - The date to which the discount offer cancellation is backdated is not prior to the G/L posting date.
   - The date to which the discount offer cancellation is backdated is not prior to the account or service's effective date.
   - The date to which the discount offer cancellation is backdated is not prior to the date of the last status change of the account or service.

   > **Note:**
   >
   > If you backdate the purchase of a discount offer on a cycle forward or cycle arrears event to a previous accounting cycle, the discount is applied only from the current accounting cycle.

3. If a **/config/provisioning_tag** object is associated with the discount offer, PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT runs the opcodes specified in that object to run at discount offer cancellation time.
4. Sets the purchase, cycle, and usage end dates for the **/purchased_discount** object to the cancellation date.
5. If you set a discount offer validity rule, it sets the PIN_FLD_USAGE_END_T and PIN_FLD_CYCLE_END_T fields to one of the following:
   - The first day of the next accounting cycle.
   - The discount offer cancellation date.
6. Calls PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT and does one of the following:
   - Deletes the **/purchased_discount** object associated with the service or account.
   - Sets the status of the **/purchased_discount** instance to **Canceled** without deleting it.

See "Customizing Discount Offer Cancellation".

> **Note:**
>
> If the account is the parent of a discount sharing group, it also deletes, or cancels without deleting, the **/purchased_discount** object from the list of discounts shared by the account.

7. Applies cycle fees or refunds discounted cycle fee amounts.

> **Note:**
>
> If the discount offer's validity period is configured to start on first usage and has not yet been set, the discount offer has not been activated. In this case, cycle fees and refunds on discounted cycle fees are not applied.

8. Generates an **/event/billing/discount/action/cancel** event to record the cancellation.

PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT returns the POID of the **/event/billing/discount/action/cancel** event.

Validating Discount Offer Dependencies

To validate discount offer dependencies, use PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY.

Configure mutually exclusive dependencies in the **/dependency** storable class.

PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY calls other opcodes to perform these operations:

- Validate the purchase time or the billing time if specified by a flag in the input. If the opcode validates the purchase time, any purchase time discount offers are selected and validated against the **/dependency** object. If the opcode validates the billing time, any subscription, system, and shared discounts are selected and validated against the **/dependency** object.
- Validate discount-to-discount exclusion rules.
- Validate discount-to-package exclusion rules, if appropriate.

If one of the operations fails, that operation, and only that operation, cancels. PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY returns a validation error and partially creates an account.

PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY validates all the discount rules and returns the resulting discount values in the return flist if PIN_FLD_FLAGS does not get set to PIN_SUBS_FLG_RETURN_ON_FIRST_ERR in the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY input flist.

PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY returns an error message when the first error occurs if PIN_FLD_FLAGS is set to PIN_SUBS_FLG_RETURN_ON_FIRST_ERR in the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY input flist. If the validation fails, PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY returns a zero result, along with the POIDs of the discount offers or package and discount offer, whichever operation occurs first.

About Optional and Required Service Types

The PIN_FLD_TYPE value of a service determines whether the service is required or optional in an account. This value is dependent on the **validate_deal_dependencies** entry in *BRM_home***/sys/cm/pin.conf**, the Connection Manager (CM) configuration file. (*BRM_home* is the directory in which the BRM server software is installed.) This entry determines whether the bundle associated with a service is required or optional.

- If the bundle is required, PIN_FLD_TYPE is set to PIN_BILL_SERVICE_REQUIRED.
- If the bundle is optional, PIN_FLD_TYPE is set to PIN_BILL_SERVICE_OPTIONAL.

When PCM_OP_CUST_CREATE_CUSTOMER creates the services in an account, it first validates whether the bundle dependency functionality is enabled in the CM **pin.conf** file. If so, it sets the service's PIN_FLD_TYPE value accordingly.

About Closing a Required Service

To set the status of a required service to **Closed** , use the PCM_OP_CUST_UPDATE_SERVICES opcode, which in turn calls PCM_OP_CUST_SET_STATUS. Set the PIN_FLD_STATUS_FLAGS value in the input flist to PIN_STATUS_FLAG_DUE_TO_REQ_SRVC. This closes all services in the package.

> **Note:**
>
> - You cannot activate an optional service that has been closed if the required service is closed.
> - When you transition a bundle or package, the service associated with the old bundle *is* closed, and its status flag value is set to PIN_STATUS_FLAG_DUE_TO_TRANSITION. You cannot change the status of a service with this status flag value.

Customizing Snowball Discounts

Use PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT to specify the distribution of group discounts to group members. You can modify this opcode code to specify an algorithm for distributing the total group discount grant to the individual group members. For example, you can specify distribution of the group discount based on group member contribution.

PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT is not called by any opcode.

Customizing Discount Offer Cancellation

To customize how discount offers are canceled, use PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT.

By default, when you cancel a **/purchased_discount** object, you set its status to **Canceled**. In addition, you set the end date to the discount offer cancellation date. PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT allows you to specify whether to delete the **/purchased_discount** object or retain it with a status of **Canceled**.

> **Note:**
>
> When delayed usage events are expected to be loaded into BRM, BRM requires information stored in the **/purchased_discount** object to rate the delayed events. In this case, the **/purchased_discount** object should always be retained with a status of **Canceled and should not be deleted**.

You can customize the behavior of PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT by setting the PIN_FLD_ACTION field to one of these values:

- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY, which sets the status of the **/purchased_discount** object to **Canceled** and does not delete it.
- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE, which deletes the **/purchased_discount** object.
- PIN_BILL_CANCEL_PRODUCT_ACTION_DONOT_CANCEL, which stops the cancellation of the discount offer.

> **Note:**
>
> All these strings are defined in the **pin_bill.h** include file.

Managing Bundles

To manage the bundles that you offer to customers, see the following:

- How Bundles Are Purchased
- Validating Changes to Bundles
- How Bundles Are Modified
- How Bundles Are Transitioned
- Customizing Bundle Transitions
- How Bundle Dependencies Are Validated
- Validating Bundle Transitions
- How Bundles Are Canceled
- Overriding Deal Settings at Customer Level

How Bundles Are Purchased

To purchase a bundle, use PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. The charge offers and discount offers in the bundle are purchased for the account or service object specified in the input flist.

If the purchase originates in an external customer relationship management (CRM) application, the input flist contains a type-only bundle *POID* because no actual BRM bundle exists.

If the bundle was created in BRM, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY to validate bundle-to-bundle dependency rules. If charge offers and discount offers are created in an external CRM, this validation does not occur.

Note:

- If you specify a **/service** object, you purchase the bundle for that service as a service bundle. The **/service** object must belong to the account.
- If you specify a NULL **/service** object, you purchase the bundle for the **/account** object as an account bundle.
- If multiple bundles are purchased for the same service type, the value of the **/service** object's **PIN_FLD_SERVICE_ID** field identifies the service instance for the bundle.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode to validates any product specification attributes on the bundle. By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes. See "Configuring Product Specification Attributes for Pricing Components" in *PDC Creating Product Offerings* or "Defining Product Specification Attributes for Pricing Components" in *BRM Setting Up Pipeline Pricing*.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL creates an identifier for each bundle purchase. This identifier is stored in the PIN_FLD_PACKAGE_ID field in the **/purchased_product** and **/purchased_discount** objects. The identifier distinguishes charge offers and discount offers bundled as part of a bundle or a package purchase.

Note:

PIN_FLD_PACKAGE_ID is unique for every bundle, except when bundles are purchased in the same package. In this case, all the offerings in the bundles in the package have the same PACKAGE_ID. To uniquely identify such a bundle (to cancel it, for example), you must provide the **/service** object along with the PACKAGE_ID. If the bundles share the same service, you must also provide the **/deal** object.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL sets the purchase, cycle, and usage validity periods of the charge offers and discount offers in a bundle. For more information, see "Managing Purchase, Cycle, and Usage Charges in Purchased Offers".

If the bundle's charge offers or discount offers grant balances that start on first usage (when the subscriber impacts the balance for the first time), you can specify that the validity period of all balances in the bundle that start on first usage are set when one of those balances is impacted for the first time. To do this, set the PIN_FLD_GRANT_RESOURCES_AS_GROUP flag value in the PIN_FLD_FLAGS field.

If you set the bundle for on-purchase billing, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL helps to create the on-purchase bill for purchase fees for charge offers associated with the bundle.

Before you purchase a bundle, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY to validate bundle dependencies. If dependencies exist, the bundle purchase cancels.

In addition, if you set the **ValidateDiscountDependency** business parameter to **enabled**, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL checks for a mutually exclusive relationship between any discount offers packaged in the bundle and any package the account already owns. The opcode retrieves the list of subscribed discount offers and price packages and it calls PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY to validate any mutual dependencies. If the opcode detects such a relationship, the bundle purchase cancels. If no mutually exclusive relationship exists, the bundle purchase continues.

Note:

- Dependencies are not checked if the bundle originated in an external customer relationship management (CRM) application. In that case, the input flist contains a type-only **/deal** object POID because no actual BRM bundle exists.
- The opcode requires no validation if PCM_OP_SUBSCRIPTION_CANCEL_DEAL was called from PCM_OP_CUST_COMMIT_CUSTOMER or PCM_OP_CUST_MODIFY_CUSTOMER.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL checks for the PIN_FLD_MODE field under the PIN_FLD_PRODUCTS array in the input flist. If the field does not exist, the opcode queries the database for the field and adds it to the flist. The PIN_FLD_MODE field determines what happens when a customer purchases the same charge offer more than once.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT to create **/purchased_product** objects for each charge offer in the bundle.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL uses the PIN_FLD_FEE_FLAG field to control whether PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies fees.

When the bundle and all charge offers and discount offers for the bundle have been purchased, the opcode records an **/event/billing/deal/purchase** event in the database for auditing purposes.

If you successfully purchase the bundle, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL returns the **/account** POID and the POID of the **/event/billing/deal/purchase** event.

You do not purchase the bundle in the following cases:

- If the **/service** object does not belong to the **/account** object.
- If the **/account** object or **/service** object is not valid.
- If status flags are not set for provisioning.

Note:

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL updates the account's and each of the associated service's PIN_FLD_EFFECTIVE_T to the purchase date.

A closed account may purchase a bundle if the **DealPurchaseForClosedAccount** business parameter is set to **enabled**. See "Enabling Charge Offer Purchases from Closed or Inactive Accounts" in *BRM Managing Customers*.

Validating Changes to Bundles

To validate changes to bundles, use PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS. This opcode first attempts to add a service to an account. If successful, it then adds or removes bundles as needed.

> **Note:**
>
> To validate bundle-to-bundle transitions, use PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY. See "Validating Bundle Transitions".

PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS performs these operations:

1. Performs validation checks on all of the bundles.
2. Calls PCM_OP_SUBSCRIPTION_CANCEL_DEAL to remove any canceled bundles.
3. Removes the bundles to be canceled from an account's list of bundles.
4. Calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to add the new bundles to the account's list of bundles.
5. Applies proration rules to the charge offers. If any cancellation or purchasing fees are defined, they are charged.
6. If a bundle created by PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS requires a new service, calls PCM_OP_CUST_MODIFY_CUSTOMER to create that service.
7. Creates an **/event/billing/product/action/cancel** object for the canceled bundles.

How Bundles Are Modified

To modify a bundle owned by a customer, use PCM_OP_SUBSCRIPTION_CHANGE_DEAL. This opcode changes the subscription charge offers associated with a bundle for an account.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL calls PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel the subscription charge offers associated with a bundle and then calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to purchase new subscription charge offers and associates them with the bundle.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL generates two notification event objects:

- The **/event/notification/deal/change** object signifies change progress. The information in the object indicates the canceled bundle.
- The **/event/notification/deal/change_complete** object signifies change completion. The information in the object indicates the new purchased bundle.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL returns the POIDs of the event objects created by PCM_OP_SUBSCRIPTION_DEAL and PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

If cancellation or purchase transactions fail, PCM_OP_SUBSCRIPTION_CHANGE_DEAL rolls back any changes made. If any part of the transaction fails, the charge offers associated with the account are not changed.

How Bundles Are Transitioned

To transition a bundle from one account to another, use PCM_OP_SUBSCRIPTION_TRANSITION_DEAL.

PCM_OP_SUBSCRIPTION_TRANSITION_DEAL performs the following steps:

1. Calls PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL to perform any custom validation that you may have added. See "Customizing Bundle Transitions".
2. Checks the **/dependency** and **/transition** objects to make sure that the bundle transition does not violate any transition rules.
3. Checks the **/deal** object to make sure that the bundle being transitioned is not a required bundle in the associated package. If it is required, the transition is not performed.
4. Performs the transition.
5. Charges cancellation fees for the old bundle and purchase fees for the new bundle based on these flags in the **/transition** object:
   - **0** charges fees.
   - **1** waives purchase fees.
   - **2** waives cancellation fees.
   - **3** waives purchase and cancellation fees.

   > **Note:**
   >
   > All cycle fees and noncurrency balances are always prorated regardless of the charge offer's cancellation proration setting.

   > **Note:**
   >
   > PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION transitions valid services with a status of **Inactive**, but those services remain permanently inactive after the transition.

6. During the transition, calls PCM_OP_BILL_CYCLE_FORWARD and PCM_OP_BILL_CYCLE_ARREARS to prorate the cycle fees, if necessary. If the PIN_TRANS_PRO_NORMAL_ON_TRANSITION flag is set, any new charge offers prorate the last cycle fee and the first cycle fee. This flag overrides the user proration settings for charge offer purchases and cancellations.

Customizing Bundle Transitions

You can customize how to transition a bundle from one account to another:

- To get a list of bundles and charge offers available for transition, use PCM_OP_CUST_POL_TRANSITION_DEALS. Use this opcode to perform any additional

filtering of bundles before they are returned as available for transition. For example, use this opcode to limit certain bundles to customers in a specific city.

PCM_OP_CUST_POL_TRANSITION_DEALS to returns the list of bundles available for an account to transition to and the charge offers the bundles contain. This opcode takes a **/deal** object POID and transition type (usually **upgrade** or **downgrade**) as input, reads the **/transition** object, and returns the list of bundles available for transition, including their charge offer names and descriptions.

Use PCM_OP_CUST_POL_TRANSITION_DEALS to perform any additional filtering of bundles before they are returned as available for transition. For example, use this opcode to limit certain bundles to customers in a specific city.

PCM_OP_CUST_POL_TRANSITION_DEALS is not called by any opcode.

- To validate how bundles are transitioned, use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN. This opcode enables customized validation based on account data during bundle-to-bundle transitions.

  Use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN to perform any additional filtering of packages before they are returned as available for transition. For example, you can use this opcode to limit certain packages to customers in a specific city.

  For example, you may restrict a bundle transition to customers from a particular location or require that customers own the first bundle for a specific period of time before allowing the transition to a different bundle.

  PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL is called by PCM_OP_SUBSCRIPTION_TRANSITION_DEAL before it performs any validation checks. PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL is an empty hook. Usually, customization consists of transition rules based on account data.

> **Note:**
>
> To customize how to transition a *package* from one account to another, use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN. See "Customizing Package Transitions without Configuring /transition Objects".

How Bundle Dependencies Are Validated

You use PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY to validate bundle dependencies.

This opcode is called by PCM_OP_CUST_SET_STATUS.

There are two ways to run PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY:

- Pass in an account and a list of bundles.

  PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY checks whether owning the bundles in the account and the bundles you send would violate any dependency rules. You can also send in an account and a package, and this opcode performs the validation checks for the bundles in the account and the bundles in the package.

- Pass in a list of bundles to validate against another list of bundles.

  PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY validates that owning both sets of bundles would not violate any dependency rules.

In the first case, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY accepts the POID for an account and validates that the bundles in that account do not violate any bundle dependency rules set in the **/dependency** object. You can also send in a **/plan** object, and PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY runs the validation checks between the bundles in the **/account** and the bundles in the package.

If any bundle violates a dependency relationship, the operation fails.

To validate two sets of bundles, send in **/account -1** (without a POID); PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY validates the bundles. If any of the bundles violates a dependency relationship, the operation fails.

Validating Bundle Transitions

To validate bundle-to-bundle dependency rules, use PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY. This opcode is called by PCM_OP_CUST_SET_STATUS.

PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY performs the following steps:

1. Confirms that the **validate_deal_dependencies** entry in the CM **pin.conf** file exists and, if so, performs validations; if not, it does nothing.
2. Checks for a **PIN_TRANS_NAME_DEAL_VALIDATION** entry in the input flist. If this entry exists, the validation checks have already been performed in the current transaction.
3. Verifies that the input is a list of bundles. If **PIN_FLD_DELETED_FLAG** is set to **1**, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY performs the validations required if those bundles were already deleted.

If the bundles pass all validation checks, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns a PIN_FLD_RESULT value of **True**. If any bundle violates any of the validation checks, this opcode fails and returns a PIN_FLD_RESULT value of **False**.

If the validation fails because the two bundles are mutually exclusive, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns the POIDs of the two bundles and a message indicating that the two bundles are mutually exclusive.

If the validation fails because a prerequisite bundle is missing, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns the POID of the bundle missing a prerequisite.

How Bundles Are Canceled

To cancel a bundle, use PCM_OP_SUBSCRIPTION_CANCEL_DEAL. This opcode cancels all charge offers and discount offers associated with the specific bundle and then cancels the bundle itself. If more than one instance of a bundle has been purchased for the account or service, each bundle instance must be canceled individually.

PCM_OP_SUBSCRIPTION_CANCEL_DEAL is called when a bundle is canceled. This opcode cancels all charge offers and discount offers associated with the specific bundle and then cancels the bundle itself.

To cancel a bundle, PCM_OP_SUBSCRIPTION_CANCEL_DEAL does the following:

1. Opens a transaction.
2. Checks if the bundle is required in the corresponding package. If so, the cancellation is not permitted.

3. From the input flist, retrieves the PIN_FLD_PACKAGE_ID, PIN_FLD_SERVICE_OBJ, name, and type of the charge offers and discount offers associated with the bundle.

   If the bundle being canceled has the same PIN_FLD_PACKAGE_ID and PIN_FLD_SERVICE_OBJ as another bundle, then PIN_FLD_DEAL_OBJ is also required to uniquely identify the bundle.
4. Validates that the bundle is available for cancellation.
5. For each charge offer, PCM_OP_SUBSCRIPTION_CANCEL_DEAL calls PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT to cancel the charge offer.
6. For each discount offer, PCM_OP_SUBSCRIPTION_CANCEL_DEAL calls PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT to cancel the discount offer.
7. PCM_OP_SUBSCRIPTION_PURCHASE_DEAL updates the account's and each of the associated service's PIN_FLD_EFFECTIVE_T to the cancellation date.
8. Creates an **/event/billing/deal/cancel** object for auditing purposes.
9. If the bundle cancel is successful, returns the **/account** POID and the POID of the **/event/billing/deal/cancel** event.

Overriding Deal Settings at Customer Level

When a customer purchases a deal, you can optionally customize one or more products in the deal. To customize a product in a deal, see the following:

- Overriding Add-On Product Validity Dates in a Deal
- Overriding Bundle Proration Settings at Customer Level

Overriding Add-On Product Validity Dates in a Deal

By default, the validity dates for any add-on product in a deal are set during design time using **loadpricelist**, **ImportExportPricing**, PCM_OP_PRICE_SET_PRICE_LIST, or PDC. However, you can override this setting for a specific customer when they purchase a deal by using the following opcodes:

- PCM_OP_SUBSCRIPTION_PURCHASE_DEAL
- PCM_OP_CUST_COMMIT_CUSTOMER
- PCM_OP_CUST_MODIFY_CUSTOMER

To override an add-on product's validity dates, set the opcode's PIN_FLD_VALIDITY_ALIGN_MODE input flist field (under PIN_FLD_DEAL_INFO.PIN_FLD_PRODUCTS) to one of the following:

- **7**: Aligns the validity dates with the specified base product. If set to **7**, you must also set the PIN_FLD_BASE_PRODUCT_OBJ input flist field to the POID of the base product on which to align validity dates.
- **8**: Aligns the validity dates with the active base charge offer that expires first.
- **9**: Aligns the validity dates with the active base charge offer that expires last.
- **10**: Aligns the validity dates with the active charge offer that expires first.
- **11**: Aligns the validity dates with the active charge offer that expires last.

This shows a sample PCM_OP_SUBSCRIPTION_PURCHASE_DEAL input flist for aligning the validity date with a specified base product:

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 113516 0
0 PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.1 /service/ip 192312 8
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DEAL_INFO   SUBSTRUCT [0] allocated 20, used 7
1    PIN_FLD_PRODUCTS        ARRAY [0] allocated 26, used 26
2       PIN_FLD_CYCLE_START_T   TSTAMP [0] (0) <null>
        ...
2       PIN_VALIDITY_ALIGN_MODE    INT [0] 7
2       PIN_FLD_BASE_PRODUCT_OBJ  POID [0] 0.0.0.1 /purchased_product 542334
```

Overriding Bundle Proration Settings at Customer Level

By default, the proration setting for each product in a deal is set during the design time using **loadpricelist**, **ImportExportPricing**, PCM_OP_PRICE_SET_PRICE_LIST, or PDC. However, you can override this setting for a specific customer when they purchase a deal by using the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL or PCM_OP_CUST_COMMIT_CUSTOMER opcode.

To do so, set the opcode's PIN_FLD_FLAGS input flist field (under PIN_FLD_DEAL_INFO.PIN_FLD_PRODUCTS) to one of the following:

- **0**: Prorated cycle fees are calculated according to the system-wide setting in your CM **pin.conf** file. See "Enabling 30-Day-Based Proration" in *BRM Configuring and Running Billing*.
- **1**: Prorated cycle fees are calculated based on a 30-day month, regardless of the number of days in the billing cycle. For example, if a deal was owned for 6 days in a cycle, the prorated fee would be the cycle fee multiplied by 0.20 (6 ÷ 30).
- **2**: Prorated cycle fees are calculated based on the actual number of days in a particular month, such as 28 days in February, 31 days in March, and 30 days in April. For example, if a deal was owned for 6 days in March, the prorated fee would be the cycle fee multiplied by 0.19 (6 ÷ 31).

This shows a sample PCM_OP_SUBSCRIPTION_PURCHASE_DEAL input flist for prorating a customer's product using a 30-day month:

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /account 101267 7
0 PIN_FLD_SERVICE_OBJ POID [0] 0.0.0.1 /service/ip 98643 6
0 PIN_FLD_PROGRAM_NAME STR [0] "Customer Center"
0 PIN_FLD_DEAL_INFO SUBSTRUCT [0] allocated 20, used 7
1    PIN_FLD_PRODUCTS ARRAY [0] allocated 35, used 35
2       PIN_FLD_FLAGS INT [0] 1
```

Managing Packages

See the following topics:

- How BRM Transitions Accounts from Source Packages to Target Packages
- Customizing Package Transitions without Configuring /transition Objects
- About Providing Transition Rules Using Policy Opcodes
- Customizing Account Bundles during Package Transitions

How BRM Transitions Accounts from Source Packages to Target Packages

BRM uses PCM_OP_SUBSCRIPTION_TRANSITION_PLAN to transition accounts from the current (source) package to a specified (target) package.

This opcode takes as input a source package that specifies the package currently owned by the account and a target package that specifies the package to transition to.

You provide the required transition type as the value of PIN_FLD_TRANSITION_TYPE in the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN input flist. The transition type can be defined as one of the following:

- **(0)**: Undefined
- **(1)**: Upgrade
- **(2)**: Downgrade
- **(3)**: Generation change
- Custom transition type

PCM_OP_SUBSCRIPTION_TRANSITION_PLAN transitions accounts from source packages to target packages in the following way:

1. Calls PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN to perform any custom validations.

   When PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN completes its actions, it passes the values provided by you to PCM_OP_SUBSCRIPTION_TRANSITION_PLAN.

2. For package transitions where there is a **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN checks the **/dependency** and **/transition** objects to make sure that the transition does not violate any transition rules.

   For package transitions where there is no **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the value supplied by you in the output flist from PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN.

3. Validates the Primary Basic Service (PBS) that must be applied on the package transition. For package transitions that have no **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the value you specify in the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN output flist.

   - For a regular transition, the two transition packages share a primary service. If not, the transaction fails.
   - For a package transition that involves a generation change, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN skips this step because the two packages do not have to share the same primary service.

4. Handles the noncurrency grants in the source package based on the value of PIN_FLD_FLAGS input field. If the value of FLD_FLAGS is PIN_SUBS_TRANSITION_CONTROL_ROLLOVER, the opcode calls a rollover function to ensure, if necessary, that the source package retains noncurrency grants and they are valid to the end of the cycle.

   To do so, the rollover function checks the package list for the source and target packages and takes the following action:

   - If the packages are in the same package list, the rollover function searches for all the sub-balances that are truncated because of the transition and extends the sub-balances to the original date before the truncation.

- If the packages are *not* in the same package list, this function searches for all the sub-balance buckets that are impacted by the transition and truncates the bucket to the current time.

5. Acts on the services in the package-to-package transition. If any services in the packages are listed as inactive, the transition is canceled.
   - If PIN_FLD_FROM_SERVICE is **NULL** or not specified in the flist, the new service in PIN_FLD_TO_SERVICE is added to the account.
   - If PIN_FLD_TO_SERVICE is **NULL** or not specified, the service specified in PIN_FLD_FROM_SERVICE is closed.
   - If PIN_FLD_FROM_SERVICE and PIN_FLD_TO_SERVICE have non-null values, only the bundles are canceled and the bundles from PIN_FLD_TO_SERVICE are purchased. Any account bundles in FROM_PLAN are canceled. Any account bundles in TO_PLAN are purchased.
6. For the package being phased out due to a generation change in the transition, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN creates a **/schedule** object that closes the services associated with the package at 00:00:00 hours at the end of the day of transition.

   Additionally, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN performs these steps:
   - Prepares the PCM_OP_CUST_UPDATE_SERVICES input flist by specifying the value of END_T as 00:00:00 hours on the following day. For example, if the transition is issued on January 15, 2004, at 12:30:00, the package being phased out ends at January 16, 2004, at 00:00:00 hours.
   - Calls PCM_OP_ACT_SCHEDULE_CREATE with the preceding input flist and the PCM_OP_CUST_MODIFY_CUSTOMER opcode number.
   - On the server side, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN

     – Provides an option that customizes the end dates, enabling the end date to occur one day later.

     – Enables a Product-End-Delay-for-Package-Transition delay configuration, which is used if the end dates are not passed. This delay configuration is then applied to the date and time of transition as an offset value.

   > **Note:**
   >
   > ERA data is not transferred.

7. Charges any cancellation fees for the old bundle and purchase fees for the new bundle based on the PIN_FLD_FEE flag (in the **/transition** object or the custom value provided by you in the output flist from PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN).

   > **Note:**
   >
   > All cycle fees and noncurrency balances are always prorated regardless of the charge offer's cancellation proration setting.

8. Creates **/event/notification/plan/transition** and **event/notification/plan/transition_complete** notification events. These events are not persistent.

BRM returns the following messages based on the success of failure of the transition:

- If the transition is successful, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns a message confirming the success.
- If the transition type is not a generation change and involves two packages that do not share a primary basic service, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns the message **The PBS does not match from-plan and to-plan**.
- If the transition violates any transition rules, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns the message **Plan is not transitionable.**

Transition Type API Considerations

The default transition types and your custom transition types are stored in the **/config/transition_type** object in the PIN_FLD_TRANSITIONS array. For example:

```
0 PIN_FLD_TRANSITIONS       ARRAY [1]
1   PIN_FLD_TYPE             ENUM [0] 1
1   PIN_FLD_TYPE_STR           STR [0] "Upgrade"
0 PIN_FLD_TRANSITIONS       ARRAY [2]
1   PIN_FLD_TYPE             ENUM [0] 2
1   PIN_FLD_TYPE_STR           STR [0] "Downgrade"
0 PIN_FLD_TRANSITIONS       ARRAY [3]
1   PIN_FLD_TYPE             ENUM [0] 3
1   PIN_FLD_TYPE_STR           STR [0] "Generation Change"
0 PIN_FLD_TRANSITIONS       ARRAY [101]
1   PIN_FLD_TYPE             ENUM [0] 101
1   PIN_FLD_TYPE_STR           STR [0] "RED"
```

The example shows these transition types:

- Upgrade - number 1
- Downgrade - number 2
- Generation Change - number 3
- RED - number 101

> **Note:**
>
> The element ID of the array is the same as its PIN_FLD_TYPE.

You could customize a policy opcode to read this object for validation checks.

Customizing Package Transitions without Configuring /transition Objects

You can customize package transitions without configuring the **/transition** object by doing the following:

- **Obtain a list of packages available for transition**: Use PCM_OP_CUST_POL_TRANSITION_PLANS.

  This opcode takes a package POID and transition type (usually upgrade or downgrade) as input, reads the **/transition** object, and returns the list of packages available for transition as output.

- **Customize the validation as necessary**: This is optional. Use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN to perform any additional validation checks or filters for transitioning packages.

  For example, you can restrict a package transition to customers from a particular location or require that customers own the first package for a specific period of time before allowing the transition to a different package.

- **Provide custom values for service and fees**: This action is required if you package to customize package transitions *without* configuring the **/transition** object.

  Modify the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN *output* flist to specify the service that must be associated with the packages and whether to waive purchase and cancellation fees associated with the packages. To do so:

  1. Add the PIN_FLD_RESULTS array to the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN *output* flist:

     ```
     0 array RESULTS
     1 int FEE_FLAG
     1 str PERMITTED
     ```

  2. Set the PIN_FLD_FEE_FLAG field in PIN_FLD_RESULTS to one of the values in Table 22-3:

     Table 22-3 PIN_FLD_FEE_FLAG Values

Value	Description
0	Charge cancellation fees for the old charge offer and purchase fees for the new charge offer.
1	Waive purchase fees.
2	Waive cancellation fees.
3	Waive purchase fees and cancellation fees.

  3. Set the PIN_FLD_PERMITTED field of this array to the Primary Basic Service (PBS) that must apply to the package transition.

  For example,

  ```
  0 PIN_FLD_RESULTS ARRAY [0] allocated 20, used 3
  1 PIN_FLD_FEE_FLAG INT [0] 3
  1 PIN_FLD_PERMITTED STR [0] "/service/telco/gsm/telephony"
  ```

PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN is called by PCM_OP_SUBSCRIPTION_TRANSITION_PLAN.

About Providing Transition Rules Using Policy Opcodes

You use PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN to automatically enable package transitions to any package without **/transition** objects.

> **Note:**
>
> Customize PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN for the package transitions for which a **/transition** object is *not* defined in PDC or Pricing Center.

When BRM attempts to transition an account and the package transition rule does not have a **/transition** object, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the following information you provide in the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN *output* flist:

- Your specification on whether to charge any cancellation fees associated with the source package for the account and purchase fees for the new charge offer for this transition.
- The Primary Basic Service (PBS) that must apply to the package transition.

> **Note:**
>
> If you do not provide these values, the opcode searches for a **/transition** object. If no **/transition** object exists for the transition rule, the transition fails.

You can provide these values to handle some package transitions and configure package transition rules in PDC or Pricing Center for others.

> **Note:**
>
> - You can customize *package* transitions in this way if the transition does *not* involve a generation change.
> - You cannot customize *bundle* transitions in this manner. PDC or Pricing Center must be used to set up transition rules to transition bundles.
> - You cannot transition packages under certain conditions. For example, you cannot transition a package when add-on bundles that are not part of the original package are active and owned by the account.

Customizing Account Bundles during Package Transitions

You can customize account bundles during package transitions. To do so, provide the associated charge offer information in the PIN_FLD_PRODUCTS input field for PCM_OP_SUBSCRIPTION_TRANSITION_PLAN. PIN_FLD_PRODUCTS is located in the PIN_FLD_DEAL_INFO substructure of the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN input flist.

Managing Contracts

To manage your customer contracts, see the following:

- Creating Contracts
- Modifying Contracts
- Canceling Contracts
- Renewing Contracts
- Extending Subscriber Contracts

Creating Contracts

To create a contract for a customer, use the PCM_OP_CONTRACT_CREATE_CONTRACT opcode. This opcode creates a **/subscriber_contract** object and generates a contract creation event. This opcode is called by Billing Care.

You can customize how contracts are created by using these Contract FM policy opcodes:

- PCM_OP_CONTRACT_POL_POST_CREATE_CONTRACT. By default, this opcode does nothing.
- PCM_OP_CONTRACT_POL_PREP_CONTRACT. By default, this opcode does nothing.
- PCM_OP_CONTRACT_POL_VALID_CONTRACT. By default, this policy opcode validates a customer's contract for valid start and end dates, pricing objects, account objects, and contract objects (during modification).

Modifying Contracts

To modify a contract owned by a customer, use PCM_OP_CONTRACT_MODIFY_CONTRACT. This opcode validates the contract input, updates the existing contract, and generates a contract modification event. This opcode is called by Billing Care.

> **Note:**
>
> You cannot modify future-dated and backdated contracts.

You can customize how contracts are modified by using the PCM_OP_CONTRACT_POL_VALID_CONTRACT policy opcode.

Canceling Contracts

To cancel a customer's contract, use PCM_OP_CONTRACT_CANCEL_CONTRACT. This opcode is called by the **pin_contracts** utility. See "pin_contracts" in *BRM Managing Customers*.

PCM_OP_CONTRACT_CANCEL_CONTRACT cancels an existing contract, cancels all pricing-associated objects, generates a contract cancellation event, and applies a penalty fee if required by the terms.

You can customize how penalty charges are applied during contract cancellation using the PCM_OP_CONTRACT_POL_CANCEL_CONTRACT policy opcode.

Renewing Contracts

To renew a customer's contract, use PCM_OP_CONTRACT_RENEW_CONTRACT. This opcode renews customer contracts that expire today.

This opcode is called by the **pin_contracts** utility.

PCM_OP_CONTRACT_RENEW_CONTRACT does the following:

1. Validates that the contract for the specified account has auto-renewal enabled, expires tomorrow, and renews the same package to new or different terms. If validation fails, renewal is not permitted.
2. If purchasing a new package, calls PCM_OP_SUBSCRIPTION_TRANSITION_PLAN to transition the specified account from the source package to the target package.
3. If purchasing a new bundle, calls PCM_OP_SUBSCRIPTION_TRANSITION_DEAL to transition the specified account from the source bundle to the target bundle.
4. Calls the general ledger module to calculate the standalone selling price. See "About the Standalone Selling Price" in *BRM Collecting General Ledger Data* for more information.
5. Returns the POIDs of the account and new package terms.

You can customize how contracts are renewed by using the PCM_OP_CONTRACT_POL_VALID_CONTRACT policy opcode.

Extending Subscriber Contracts

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

You can extend the **/subscriber_contract** storable class to store additional fields by customizing the PCM_OP_CONTRACT_POL_PREP_CONTRACT policy opcode. Customize the policy opcode to add the custom fields to its PIN_FLD_INHERITED_INFO output flist substruct, which is returned to the PCM_OP_CONTRACT_CREATE_CONTRACT opcode during contract creation.

After you extend the **/subscriber_contract** storable class, you can populate data in the additional fields by passing the following input flist fields to the PCM_OP_CUST_COMMIT_CUSTOMER, PCM_OP_CUST_MODIFY_CUSTOMER, and PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcodes:

- Use PIN_FLD_CONTRACT_INFO.PIN_FLD_POID to pass in the type-only POID of the subscriber contract object to create.
- Use PIN_FLD_CONTRACT_INFO.PIN_FLD_INHERITED_INFO to pass data to the additional fields in the **/subscriber_contract** object.

For example, this shows how to pass in a bearer service and IMEI number to a **/subscriber_contract/gsm** object using PCM_OP_CUST_COMMIT_CUSTOMER:

```
0 PIN_FLD_CONTRACT_INFO         SUBSTRUCT [0] allocated 20, used 7
1   PIN_FLD_POID                     POID [0] 0.0.0.1 /
subscriber_contract/gsm -1 0
1     PIN_FLD_INHERITED_INFO    SUBSTRUCT [0] allocated 20, used 1
2     PIN_FLD_GSM_INFO          SUBSTRUCT [0] allocated 20, used 2
3        PIN_FLD_BEARER_SERVICE       STR [0] "test_custom"
3        PIN_FLD_IMEI                 STR [0] "1234"
```

Managing Purchase, Cycle, and Usage Charges in Purchased Offers

Purchase, cycle, and usage validity periods define when charge offers and discount offers are valid and when charge offer fees are charged and discounted.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT, which set the purchase, cycle, and usage validity periods of the bundle's charge offers and discount offers, respectively.

If the system is configured for time-stamp rounding, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT round the start and end times to 00:00:00 hours. However, when a charge offer's purchase, cycle, and usage start and end units are set to 1 (seconds), 2 (minutes), or 3 (hours), and the validity period is less than 24 hours, time stamps are not rounded, regardless of your system configuration. If the validity period is greater than 24 hours, the cycle end time stamp is rounded when calculating the scale to determine the cycle fee amount to charge.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL uses the purchase, cycle, and usage validity dates that are specified in the **/deal** object by default. If you specify alternative validity periods for charge offers and discount offers, those validity dates permanently override the dates specified in the **/deal** object.

To set the purchase, cycle, and usage start and end times, pass the following values in opcode input flists in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays. In the following fields, the asterisk (*) indicates either PURCHASE, CYCLE, or USAGE.

> **Note:**
>
> The cycle and usage validity periods must fall entirely within the purchase validity period.

- Specify the start time as follows:
  - To start immediately (when the charge offer or discount offer is purchased), set the value of PIN_FLD_*_START_T and PIN_FLD_*_START_UNIT to **0**. The opcode sets the start time to the purchase time.
  - To start on first usage (when the subscriber uses the charge offer or discount offer for the first time) set the PIN_FLD_*_START_UNIT field to **-1** and the PIN_FLD_*_START_OFFSET field to **0**.
  - To start relative to the purchase time, set the relative unit (for example, days or cycles) in PIN_FLD_*_START_UNIT and set the number of relative units in PIN_FLD_*_START_OFFSET. The relative start time is calculated by adding the relative offset period to the purchase time.
- Specify the end time as follows:
  - To never end, set the value of PIN_FLD_*_END_T and PIN_FLD_*_END_UNIT to **0**.
  - To end relative to the start time, set the relative unit (for example, days or cycles) in PIN_FLD_*_END_UNIT and set the number of relative units in PIN_FLD_*_END_OFFSET. The relative end time is calculated by adding the relative offset period to the start time.

When the charge offer or discount offer is purchased, the start and end times are calculated based on the purchase time. The validity period is set in the PIN_FLD_*_START_T and PIN_FLD_*_END_T fields of the account's **/purchased_product** and **/purchased_discount** objects.

If the validity period has a relative start or end time, the details about the relative offset are stored in the PIN_FLD_*_START_DETAILS and PIN_FLD_*_END_DETAILS fields of the purchased charge offer or purchased discount offer. The details fields store three values: the mode of the validity period, the relative offset unit, and the number of offset units in the relative period. For more information, see "Storing Relative Start and End Times for an Account's Charge Offers and Discount Offers".

Changing the Purchase, Usage, and Cycle Start and End Times

PCM_OP_SUBSCRIPTION_SET_PRODINFO is called to change the purchase, cycle, or usage start and end times of an account's purchased charge offer. This opcode is called by:

- PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS when the status of an account's charge offer is changed.
- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT when the same charge offer is purchased multiple times and the charge offer's PIN_FLD_MODE field is set to **1** or **2**.

If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_SUBSCRIPTION_SET_PRODINFO returns the entire event flist for the events created as a result of the modification. If the flag is not set, the opcode returns the event POIDs of all event objects created as a result of the modification.

After a charge offer is purchased and its start and end dates have been set, you can change the start time to specify only another fixed date, and you can change the end time to specify only another fixed date or to end never.

To change the purchase, cycle, or usage validity period, specify the new start and end dates in their respective START_T and END_T fields in the PIN_FLD_PRODUCTS array in the input flist.

If you change the purchase, cycle, or usage start time from first usage to a fixed date, PCM_OP_SUBSCRIPTION_SET_PRODINFO generates an event that causes the validity period to be updated in ECE.

The start and end dates are stored in the account's **/purchased_product** object.

If your system is configured to round time stamps, and if you modify the purchase, usage, or cycle start and end times, PCM_OP_SUBSCRIPTION_SET_PRODINFO rounds the new times to 00:00:00 hours. However, if the purchase, cycle, and usage start and end units are set to 1 (seconds), 2 (minutes), or 3 (hours), and the validity period is less than 24 hours, time stamps are not rounded, regardless of your system configuration.

BRM does not allow you to change a charge offer's purchase, usage, and cycle start and end dates as follows:

- The charge offer purchase start date if the purchase start date has elapsed.
  For example, on January 1, a charge offer is purchased with the purchase start date set to January 15. On January 10, you can backdate the purchase start date to January 5. But, you cannot backdate the purchase start date on January 16; that is, you cannot backdate the purchase date when the January 15 date has passed.
- The charge offer cycle start date if the cycle fees have been applied.
- The charge offer purchase, usage, and cycle end date prior to the respective purchase start date.
- The charge offer purchase, usage, cycle start or end dates prior to the G/L posting date.

See "About Backdated Purchase, Usage, or Cycle End Dates".

Overriding Charges and Discounts for a Period of Time

Use PCM_OP_SUBSCRIPTION_CRUD_OFFER_OVERRIDE to specify when and how to override charges and discounts during specified time periods. It creates, reads, updates, or deletes a specified **/offering_override_value** object based on the value of an input flag and validates specified price tags against the price tag configuration objects.

For more information, see "Dynamically Changing One-Time and Recurring Fees Based on Date" in *BRM Setting Up Pipeline Pricing*.

When calling the opcode, pass in the following input data:

- The POID of the **/offering_override_values** object.
- The POID of the **/purchased_product** or **/purchased_discount** object.
- The operation to perform on the **/offering_override_values** object.

To specify the operation to perform on the **/offering_override_values** object, set the PIN_FLD_FLAG field in the PIN_FLD_OVERRIDE_PRODUCTS or PIN_FLD_OVERRIDE_DISCOUNTS array to one of the following:

- **1**: Adds a date range and price tag combination for the specified event type. That is, it adds a PIN_FLD_OVERRIDE_DATE_RANGE buffer to each matching PIN_FLD_USAGE_MAP array.

  Calls PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS to check that the price tags in the PIN_FLD_PRICE_TAGS array are valid when compared to the price tag configuration objects. For products, the product POID is used to compare with only the price tags permitted for that product. For discounts, all tags are compared.
- **2**: Adds or updates the price tag name, amount or percentage, and (optional) resource ID for the specified date ranges.

  Calls PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS to check that the price tags in the PIN_FLD_PRICE_TAGS array are valid when compared to the price tag configuration objects. For products, the product POID is used to compare with only the price tags permitted for that product. For discounts, all tags are compared.
- **3**: Deletes the specified **/offering_override_values** object.
- **4**: Deletes all expired date range and pricing tag combinations from the **/offering_override_values** object. That is, it deletes from the PIN_FLD_USAGE_MAP array all PIN_FLD_DATE_RANGES buffers with expired date ranges.
- **5**: Deletes the specified pricing tag from the **/offering_override_values** object's PIN_FLD_DATE_RANGES buffer.
- **6**: Reads the **/offering_override_values** object.
- **8**: Deletes all usage maps for the specified event type. For example, it deletes all PIN_FLD_USAGE_MAPS arrays associated with the **/event/billing/product/fee/cycle/cycle_forward_monthly** event.

PCM_OP_SUBSCRIPTION_CRUD_OFFER_OVERRIDE is called by the following opcodes when creating products and discounts:

- PCM_OP_SUBSCRIPTION_SET_PRODINFO when the PIN_FLD_OVERRIDE_PRODUCTS array is passed in the PIN_FLD_PRODUCTS array.
- PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO when the PIN_FLD_OVERRIDE_DISCOUNTS array is passed in the PIN_FLD_DISCOUNTS array.

Your third-party client applications can also call PCM_OP_SUBSCRIPTION_CRUD_OFFER_OVERRIDE directly.

Specifying Purchase, Cycle, and Usage Start and End Times

When the purchase, cycle, or usage period has a fixed start time (starts and ends on a specific date and time), the start and end times are passed in opcode flists in time stamp fields:

- PIN_FLD_PURCHASE_START_T
- PIN_FLD_PURCHASE_END_T
- PIN_FLD_CYCLE_START_T
- PIN_FLD_CYCLE_END_T
- PIN_FLD_USAGE_START_T
- PIN_FLD_USAGE_END_T

When the purchase, cycle, or usage period has a relative start or end time, the details about the relative period are passed in opcode flists in *unit* and *offset* fields:

- These unit fields specify the type of relative unit:
  - PIN_FLD_PURCHASE_START_UNIT
  - PIN_FLD_PURCHASE_END_UNIT
  - PIN_FLD_CYCLE_START_UNIT
  - PIN_FLD_CYCLE_END_UNIT
  - PIN_FLD_USAGE_START_UNIT
  - PIN_FLD_USAGE_END_UNIT

  The unit can be one of these values:
  - Seconds = 1
  - Minutes = 2
  - Hours = 3
  - Days = 4
  - Accounting cycles = 8
- These offset fields specify the number of units in the relative period:
  - PIN_FLD_PURCHASE_START_OFFSET
  - PIN_FLD_PURCHASE_END_OFFSET
  - PIN_FLD_CYCLE_START_OFFSET
  - PIN_FLD_CYCLE_END_OFFSET
  - PIN_FLD_USAGE_START_OFFSET
  - PIN_FLD_USAGE_END_OFFSET

The unit and offset fields are used in combination to determine the relative offset of the purchase, cycle, and usage periods. For example:

- If a charge offer's PIN_FLD_CYCLE_START_UNIT has a value of **8** (accounting cycles) and PIN_FLD_CYCLE_START_OFFSET has a value of **3**, the cycle fee starts three accounting cycles after the charge offer is purchased.

- If a discount offer's PIN_FLD_PURCHASE_END_UNIT has a value of **8** (accounting cycles) and PIN_FLD_PURCHASE_END_OFFSET has a value of **3**, the discount offer expires three accounting cycles after it is activated.

You can specify any number of units in the offset fields, up to 1048576. This is equivalent to approximately 12 days in seconds, 728 days in minutes, and 119 years in hours.

The relative offset information in the unit and offset fields is stored in the BRM database in *details* fields. For more information, see "Storing Relative Start and End Times for an Account's Charge Offers and Discount Offers".

Calculating the Cycle Forward Fee

If you modify the cycle start time or cycle end time for the charge offer in the middle of a cycle, PCM_OP_SUBSCRIPTION_SET_PRODINFO calculates the cycle forward fee or cycle arrears fee for the charge offer.

If the charge offer has a cycle forward fee, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD is called to perform the following actions:

- If you change the cycle start time from a future time to the current time, applies the cycle forward fee for the current time through the end of the current cycle.
- If you change the cycle end time to an earlier time and the cycle forward fee is already applied for that cycle, refunds the cycle forward fee for the unused portion of the cycle.
- If you change the cycle end time to a later time and the cycle forward fee is already applied for that cycle, applies the cycle forward fee for the end of the old cycle end time through the end of the new cycle.

Calculating the Cycle Arrears Fee

If you modify the cycle start time or cycle end time for the charge offer in the middle of a cycle, PCM_OP_SUBSCRIPTION_SET_PRODINFO calculates the cycle forward fee or cycle arrears fee for the charge offer.

> **Note:**
>
> - Refunds for cycle arrears fees are not supported. Use caution when changing cycle end times.
> - Changing the cycle end time for a cycle arrears fee has some limitations. See "Cycle Arrears Charge Offer Limitations".

If the charge offer has a cycle arrears fee, PCM_OP_SUBSCRIPTION_CYCLE_ARREARS is called to perform the following actions:

- If you change the cycle end time from a future time to the current time, the cycle arrears fee is applied for the cycle.
- If you change the cycle end time from a future time to an earlier time, but later than the current time, the cycle arrears fee is *not* applied. However, the cycle arrears fee is applied during the next billing run.

Cycle Arrears Charge Offer Limitations

If you change the cycle end time *only once* in a cycle, the cycle fee is applied either the day of the change or the next billing day, depending on the new cycle end value. If the new PIN_FLD_CYCLE_END_T value is the same as the event time, the cycle arrears fee is charged at the same time; otherwise, it is charged at the next billing time.

> **Note:**
>
> If you use time stamp rounding, the cycle end time is compared to the event date. Otherwise, the cycle end time is compared to the event time (the current time).

If you change the cycle end date *more than once* in a cycle:

- If the cycle arrears fee from the first change has not yet been applied, it will not apply after the second change is made.
- If the new cycle end date is changed to **0** (never) or to a date later than the end of the current accounting cycle, cycle fees apply accordingly, depending on which of the following cancellation settings you specified when setting up proration for your pricing package:
  - **Charge full**: Applies no cycle fees for the whole cycle.
  - **Based on usage and is proratable**: Applies cycle fees from the new cycle end date to the end of the cycle.
  - **No Charge and proratable**: Applies cycle fees from the new cycle end date to the end of the cycle.
  - **No Charge and is not proratable**: Applies full cycle fees.

About Backdated Purchase, Usage, or Cycle End Dates

Setting a charge offer purchase, usage, or cycle start dates to a backdated date is different from backdating a purchase itself. When the charge offer purchase, usage, or cycle start dates are set to a backdated date, though the cycle charges are applied from the backdated date, the change is effective only from the current time.

For example, you purchase a charge offer on February 1, and set the PURCHASE_ START_T, USAGE_START_T, and CYCLE_START_T fields of the charge offer to a backdated date of January 15. In this case, the charges are applied from January 15, but the creation time of the charge offer remains February 1 (current time). Thus, if usage events during the period of January 15 to February 1 are rerated, they are not rated with this charge offer.

Here, if you enter January 15 as PIN_FLD _END_T in the opcode, the charge offer becomes effective from January 15 (the creation time of the charge offer is set to January 15), and the charges are also applied from the backdated date of January 15.

Oracle recommends that you use the latter procedure for backdating a purchase.

> **Note:**
>
> You cannot set the discount offer PURCHASE_ START_T, USAGE_START_T, and CYCLE_START_T to a backdated date.

Similarly, backdating a charge offer or discount offer's purchase, usage, or cycle end dates is different from backdating the cancellation. When the charge offer or discount offer's purchase, usage, or cycle end dates are set to backdated date, though the cycle charges are refunded from the backdated date, the change is effective only from the current time.

For example, on February 1, you set the PURCHASE_ END_T, USAGE_END_T, or CYCLE_END_T fields of a charge offer to the backdated date of January 15. In this case, the charges are refunded from January 15, but the change in the END_T to January 15 is effective only from the current time (February 1). Thus, if usage events during the period of January 15 to February 1 are rerated, they are rated with this charge offer.

Here, if you enter January 15 as PIN_FLD _END_T in the opcode, the charge offer is available for rating from January 15, and the charges are also refunded from the backdated date of January 15.

BRM does not allow you to backdate the following:

- The charge offer purchase start date if the purchase start date has elapsed.

  For example, on January 1, a charge offer is purchased with the purchase start date set to January 15. On January 10, you can backdate the purchase start date to January 5. But, you cannot backdate the purchase start date on January 16; that is, you cannot backdate the purchase date when the January 15 date has passed.
- The charge offer cycle start date if the cycle start date has elapsed.
- The usage and cycle end date of a charge offer prior to its purchase start date.
- The purchase, usage, and cycle end date of a discount offer prior to its purchase start date.
- The purchase, usage, or cycle start and end dates of a charge offer prior to the G/L posting date.
- The purchase, usage, and cycle end dates of a discount offer prior to the G/L posting date.

Example of Backdated Purchase, Usage, and Cycle Start and End Dates

On April 1, consider that you create an account that owns a charge offer with $9.95 as a cycle forward fee, 3600 as Anytime Minutes, and the start date set to June 20. On May 2, you change the start date to April 16. The charges are as follows:

- $4.98 monthly charges and 1800 Anytime Minutes for the period from April 16 to May 1.
- $9.95 monthly charges and 3600 Anytime Minutes for the period from May 1 to June 1.

Backdating Purchase, Usage, or Cycle Start and End Dates

To backdate the purchase, usage, or cycle start or end dates of a charge offer or purchase, usage, or cycle end date of a discount offer during purchase, enter the backdated date in the respective date fields.

After a charge offer is purchased, use PCM_OP_SUBSCRIPTION_SET_PRODINFO to backdate the purchase, usage, or cycle start and end dates. Similarly, use PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO to backdate the purchase, usage, or cycle end dates of the discount offer.

> **Note:**
>
> If you set the end date of a discount offer on a cycle forward or cycle arrears event to a previous accounting cycle, the discount offer is refunded only for the current accounting cycle.

Storing Relative Start and End Times for an Account's Charge Offers and Discount Offers

Relative validity information in the PIN_FLD_*_UNIT and PIN_FLD_*_OFFSET fields that are passed in opcode flists are stored in the database in *details* fields:

- PIN_FLD_PURCHASE_START_DETAILS
- PIN_FLD_PURCHASE_END_DETAILS
- PIN_FLD_CYCLE_START_DETAILS
- PIN_FLD_CYCLE_END_DETAILS
- PIN_FLD_USAGE_START_DETAILS
- PIN_FLD_USAGE_END_DETAILS

When charge offers and discount offers are purchased, the details fields are stored in **/purchased_product** and **/purchased_discount** objects.

The start-details and end-details fields are 32-bit integer fields that store the mode of the validity period, the relative offset unit, and the number of offset units in the relative period:

- **Mode**: The mode specifies generally when the validity period starts or ends. The mode is stored in the lower 8th bit.

  When a charge offer or discount offer is purchased, its start and end dates are set and the mode value in the start-details field is set to PIN_VALIDITY_ABSOLUTE (a value of **0**).

  The end-details field can have a value of PIN_VALIDITY_ABSOLUTE if the end date is specified or a value of PIN_VALIDITY_NEVER (a value of **2**) if the validity period never ends.

  > **Note:**
  >
  > Before the charge offer or discount offer is purchased, the mode in the details field of the charge offer or discount offer in the **/deal** object can also specify other values such as *immediate* and *relative*. The mode helps to determine how the start and end times are set when the charge offer or discount offer is purchased.

- **Relative offset unit**: This value specifies the type of offset unit and corresponds to the value of PIN_FLD_*_START_UNIT or PIN_FLD_*_END_UNIT that is passed in opcode flists. The relative offset unit is stored in the next four bits of the details field and can have one of these values:
  - Seconds = 1
  - Minutes = 2
  - Hours = 3

  - Days = 4
  - Accounting cycles = 8

- **Number of offset units**: This value specifies how many offset units are in the relative period and corresponds to the value of PIN_FLD_*_START_OFFSET or PIN_FLD_*_END_OFFSET that is passed in opcode flists. The number of offset units is stored in the remaining 20 bits of the details field.

How Cycle Fees Are Calculated for Backdated Cancellation or Inactivation

When you backdate a cancellation or inactivation, the refunds for the events that occurred during the backdated period are prorated and applied accordingly.

PCM_OP_SUBSCRIPTION_CYCLE_FORWARD determines the scale values that are used by for prorating the cycle fee amount to be refunded. This opcode prorates and applies the cycle fee amount to be refunded when a charge offer or discount offer is canceled or inactivated.

When you backdate the cancellation of a charge offer with cycle forward events, cycle forward fees are prorated and refunded from the backdated cancellation date.

Example of backdated charge offer cancellation

When you backdate the cancellation of a charge offer with cycle forward events, cycle forward fees are prorated and refunded from the backdated cancellation date.

For example, consider that you create an account on September 1 with the DOM set to 1. On November 5, you backdate the cancellation of the charge offer with the following details to September 15:

- Cycle forward fee $3.00.
- Proration set to **Charge based on amount used**.

The three cycles including the current cycle are refunded as follows:

- $1.50 for the period of September 15 to October 1 (for 15 days).
- $3.00 for the period of October 1 to November 1.
- $3.00 for the period of November 1 to December 1.

These refunds are included in the next bill that gets generated during the bill run on December 1.

Example of backdated discount offer cancellation

On April 1, consider that you create an account that owns a charge offer with $50 as a cycle forward fee and a discount of 10% on the monthly cycle fees. A cycle fee of $50 and a discount of $5 is applied for the period of April 1 to May 1, making the balance $45.

On April 28, you cancel the discount offer backdated to April 16. The discount of $2.50 applied for the period of April 16 to May 1 is refunded, making the balance as $47.50.

How Earned and Unearned Revenue Is Set for Backdated Period

In the case of cycle forward events, the charges for the backdated period have the EARNED_START_T (**billing_cycle_start**) and EARNED_END_T (**billing_cycle_ end**) set to the backdated period.

For example, on February 15, a charge offer with cycle forward events for an account with the DOM set to 1 is purchased, backdated to January 15. The EARNED_ START_T and EARNED_END_T for the backdated period (January 15 through February 1) are January 15

and February 1 respectively, while the EARNED_START_T and EARNED_END_T for the current cycle are February 1 and March 1 respectively.

Applying Recurring Charges

The main opcodes used for applying recurring charges are:

- PCM_OP_SUBSCRIPTION_CYCLE_FORWARD. See "Applying Cycle Forward Fees" for more information.
- PCM_OP_SUBSCRIPTION_CYCLE_ARREARS. See "Applying Cycle Arrears Fees" for more information.

Applying Cycle Forward Fees

To apply a cycle forward fee to a balance group, use PCM_OP_SUBSCRIPTION_CYCLE_FORWARD.

This opcode is called to charge or refund cycle forward fees (for example, when a charge offer or discount offer is purchased, canceled, activated, or inactivated).

PCM_OP_SUBSCRIPTION_CYCLE_FORWARD performs these operations:

1. When a charge offer or discount offer is *purchased* or *activated* during the cycle, determines scale values that are used for prorating the cycle forward fee amount to be charged.
2. When a charge offer or discount offer is *canceled* or *inactivated* during the cycle, determines scale values that are used for prorating the cycle forward fee amount to be refunded.
3. Uses customized charge offers when valid to calculate fees or refunds. If a customized charge offer is valid for only part of a cycle, it contributes toward the total charge or refund based on the length of its validity.
4. When there is a pricing change scheduled for the *next* or *current* cycle, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD gets a list of prices and the period in which they are applicable and calculates the correct charges.
5. When the **ApplyRolloverBeforeCycleFees** business parameter is enabled, calls PCM_OP_SUBSCRIPTION_CYCLE_ROLLOVER to roll over unused resources from one cycle to succeeding cycles.
6. Calculates and sets CYCLE_FEE_START_T and CYCLE_FEE_END_T to the next period, if applicable.
7. After all the charge offers, discount offers, and their cycle forward fees are determined, it sends the information to PCM_OP_ACT_USAGE to rate and apply the charges.

> **Note:**
>
> If the charge offer or discount offer starts on first usage (when the customer first uses the charge offer or discount offer) and its validity period has not yet been set, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD does not call PCM_OP_ACT_USAGE and cycle fees are not applied.

8. For auditing purposes, it creates **/event/billing/product/fee/cycle/cycle_forward**_*type* objects, where *type* is the frequency of the cycle forward charge (for example, daily, weekly, monthly, bimonthly, quarterly, semiannual, or annual).
9. If successful, it returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_forward**_*type* event.

Applying Cycle Arrears Fees

To apply a cycle arrears fee, use PCM_OP_SUBSCRIPTION_CYCLE_ARREARS.

> **Note:**
>
> Cycle arrears fees are applied only for a single month.

PCM_OP_SUBSCRIPTION_CYCLE_ARREARS performs these operations:

1. When charge offers or discount offers with cycle fees are canceled or inactivated during a cycle, calculates the scale to prorate the cycle arrears fee amount to be charged.
2. When there is a pricing change in the *previous* or *next* cycle, gets a list of prices and the period in which they are applicable and calculates the correct charges.
3. Uses customized charge offers when valid to calculate fees or refunds. If a customized charge offer is valid for only part of a cycle, it contributes a portion of the total charge or refund based on the length of its validity.
4. After all the cycle arrears fees are determined, sends the information to PCM_OP_ACT_USAGE to rate and apply the charges.

> **Note:**
>
> If the charge offer or discount offer starts on first usage and its validity period has not yet been set, PCM_OP_SUBSCRIPTION_CYCLE_ARREARS does not call PCM_OP_ACT_USAGE and cycle fees are not applied.

5. Creates the **/event/billing/product/fee/cycle/cycle_arrears** event for auditing purposes.
6. If successful, returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_arrears** event.

Applying Folds

Use PCM_OP_SUBSCRIPTION_CYCLE_FOLD to apply balance impacts for fold events.

PCM_OP_SUBSCRIPTION_CYCLE_FOLD performs these operations:

1. Calls PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD to get the order in which to fold the account's balances. By default, folds are applied in ascending order of the balance element IDs.
2. Checks the value of the balance's PIN_FLD_APPLY_MODE field in the **/config/beid** object to see whether the balance is specified to be rolled over. For more information, see "Specifying Which Balances to Fold".

3. Applies folds for all of the account's balance groups and also for each balance in the balance group that is specified to be rolled over. However, if a balance element ID is specified in the input flist, only that balance is folded.
4. After applying the balance impacts, creates **/event/billing/cycle/fold** event for auditing purposes.
5. If successful, returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/fold** event.

Customizing How Folds Are Applied

To customize how folds are applied, use PCM_OP_SUBSCRIPTION_POL_PRE_FOLD.

For example, when billing is run, this opcode is called to verify that the **pin_cycle_fees** utility has applied cycle fees to an account.

This opcode is called by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT and PCM_OP_BILL_MAKE_BILL.

This opcode is an empty hook.

Customizing the Order to Apply Folds

To customize the order in which folds are applied, use PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD.

By default, balances are folded in ascending order based on the balance element ID. This opcode enables you to change the order in which balances are folded.

For example, you can fold balances in descending order of the balance element IDs. To do this, sort the PIN_FLD_BALANCES array and return the sorted array.

This example shows the PIN_FLIST_SORT statement for sorting the balances array in descending order:

```
*out_flistp = PIN_FLIST_COPY(i_flistp, ebufp);
s_flistp = PIN_FLIST_CREATE(ebufp);
PIN_FLIST_ELEM_SET(s_flistp, (void *)NULL, PIN_FLD_BALANCES, PIN_ELEMID_ANY, ebufp);

PIN_FLIST_SORT(*out_flistp, s_flistp, 1, ebufp);
```

PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD returns the contents of the input flist including the POID of the **/balance_group** object and the balance array sorted in ascending order.

PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD is called by PCM_OP_SUBSCRIPTION_CYCLE_FOLD.

Specifying Which Balances to Fold

You can specify which balances are impacted by fold events. This enables you to exclude folding balances that do not need to be folded.

To specify the balances to fold, use PDC or Pricing Center.

Customizing Which Balances to Fold When Charge Offers Are Canceled

PCM_OP_SUBSCRIPTION_POL_PREP_FOLD prepares the list of balances that must be folded when a charge offer is canceled.

This opcode is called when a charge offer is canceled and it performs the following functions:

- Checks the canceled charge offer.
- Creates a list of balances affected by the cancellation that must be folded.
- Calls PCM_OP_SUBSCRIPTION_CYCLE_FOLD and passes the list of balances as the input.

If you do not want to fold balances after a charge offer is canceled, customize PCM_OP_SUBSCRIPTION_POL_PREP_FOLD by removing or commenting out the code in the **fm_subscription_pol_prep_fold.c** file.

PCM_OP_SUBSCRIPTION_POL_PREP_FOLD is called by PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS.

Getting Data about Bundles, Charge Offers, Discount Offers, and Services

See the following topics:

- Getting Purchased Offers
- Getting Packages, Bundles, and Charge Offers for Purchase
- Getting a List of Bundles, Charge Offers, Discount Offers, and Services
- Getting a List of Packages and Bundles That an Account Owns
- Reading Data for All Valid Purchased Charge Offers and Discount Offers
- Finding Events Associated with Bundles, Charge Offers, Discount Offers, and Services

Getting Purchased Offers

PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS retrieves the purchased charge offers and discount offers associated with an account. Because charge offers and discount offers were part of the account object, reading the account object would fetch all the purchased charge offers and discount offers.

The opcode requires a scope object in the PIN_FLD_SCOPE_OBJ of the input flist. The scope object can be an account, bill unit (**/billinfo** object), or service object. The meanings of passing these different scope objects are as follows:

- **/account object:** Fetches all charge offers and discount offers for the account and its services.
- **/billinfo object:** Fetches charge offers and discount offers that contribute to the specified bill unit.
- **/service object:** Fetches charge offers and discount offers that belong to the specified service.

The opcode logic performs a main search based on the input flist passed in the input. The search is repeated if there are more result sets to be fetched. For example, if a service object is passed as the scope object, the input flist would look like this:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 618010 0
0 PIN_FLD_SCOPE_OBJ      POID [0] 0.0.0.1 /service/ip 615706 3
0 PIN_FLD_STATUS_FLAGS   INT [0] 3
0 PIN_FLD_VALIDITY_FLAGS INT [0] 3
```

```
0 PIN_FLD_INCLUSION_FLAGSINT [0] 2
0 PIN_FLD_OVERRIDE_FLAGS INT [0] 2
0 PIN_FLD_END_T          TSTAMP [0] (1154415600) Tue Aug  1 00:00:00 2006
0 PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 615702 3
0 PIN_FLD_OVERRIDDEN_OBJ  POID [0] 0.0.0.1 /purchased_product 324706 0
0 PIN_FLD_PACKAGE_ID INT [0] 23
0 PIN_FLD_PRODUCTS       ARRAY [*]     NULL array ptr
0 PIN_FLD_DISCOUNTS      ARRAY [*]     NULL array ptr
```

The input for the opcode also contains qualifiers to fetch the correct set of offerings:

- To specify whether to fetch only charge offers or only discount offers use the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays.
- To specify only a limited number of fields to fetch, use the fields under the PRODUCTS and DISCOUNTS arrays.
- To fetch charge offers/discounts valid as of a given time, the PIN_FLD_END_T field can be passed in the input. Additional qualifiers such as cycle, usage, or purchase can be passed in as PIN_FLD_VALIDITY_FLAGS.
- PIN_FLD_STATUS_FLAGS:
  - PIN_SUBS_FLG_OFFERING_STATUS_ACTIVE: Get only active offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_INACTIVE: Get only inactive offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_CLOSED: Get only closed offerings.

  > **Note:**
  >
  > Use of multiple values implies the target object should satisfy any of the above.

- PIN_FLD_VALIDITY_FLAGS:
  - PIN_SUBS_FLG_OFFERING_VALIDITY_CYCLE: Compare the END_T value with the CYCLE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_PURCHASE: Compare the END_T value with the PURCHASE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_USAGE: Compare the END_T value with the USAGE_END_T value.

  > **Note:**
  >
  > Use of multiple flags implies the target object must satisfy all of the above.

- PIN_FLD_INCLUSION_FLAGS:
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_PRODS: Include all eligible charge offers, including account and subscription charge offers.
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_DISCS: Include all eligible discount offers, including account and subscription discount offers.

> **Note:**
>
> When this field is missing, only eligible offerings from a specified scope are returned.

- PIN_FLD_OVERRIDE_FLAGS:
  - PIN_SUBS_FLG_OFFERING_ACCT_LEVEL_ONLY: Filter out only account level offerings. Valid for **/account** objects only.
  - PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY: Must be used with the OVERRIDDEN _OBJ field. When a valid offering POID is sent, this flag returns all the offerings that override the input offering. When a NULL offering POID is sent, only the base charge offers are returned. Valid for any scope.

> **Note:**
>
> When none of these are present (or this field is missing), all charge offers are returned.

- PIN_FLD_PACKAGE_ID: Limits a search to the package ID, which translates to a single package. This can be used with any scope.
- PIN_FLD_DEAL_OBJ: Limits a search to objects that are part of the same bundle by entering the bundle object POID.

Getting Packages, Bundles, and Charge Offers for Purchase

Use the following opcodes to get packages, bundles, and charge offers for customer purchase:

- To get a list of packages for customer purchase, use PCM_OP_CUST_POL_GET_PLANS. This opcode chooses the packages to return based on the Automatic Account Creation (AAC) fields in the input flist, enabling the opcode to qualify different packages depending on the values of the fields returned by each customer.

  You must customize PCM_OP_CUST_POL_GET_PLANS to search for and display the package list by customer type. By default, if you pass in a type-only POID, the opcode retrieves **new** packages; otherwise, it retrieves **addon** packages.

  You can customize PCM_OP_CUST_POL_GET_PLANS to search the **/group/plan_list** object by customer type and display the correct package lists based on the customer type.

  PCM_OP_CUST_POL_GET_PLANS is not called by any opcode.
- To get a list of bundles for customer purchase, use PCM_OP_CUST_POL_GET_DEALS. This opcode searches for all **/deal** objects that are valid (bundle **END_T = 0** is valid). Each bundle in the list is then read and checked to see whether the bundle's **permitteds** array allows the object type to purchase it.

  PCM_OP_CUST_POL_GET_DEALS is not called by any opcode.
- To get a list of charge offers for customer purchase, use PCM_OP_CUST_POL_GET_PRODUCTS. The charge offer's PIN_FLD_PERMITTEDS array is checked for valid object types purchasing the charge offer.

  PCM_OP_CUST_POL_GET_PRODUCTS is not called by any opcode.

- Use PCM_OP_CUST_POL_READ_PLAN to customize bundles during account creation. For a given package, this opcode constructs a tree of services, bundles, and charge offers associated with that package. This opcode retrieves account packages in addition to packages related to services.

  PCM_OP_CUST_POL_READ_PLAN is not called by any opcode.

Getting a List of Bundles, Charge Offers, Discount Offers, and Services

To get the hierarchical relationships of bundles, charge offers, discount offers, and services associated with an account, use PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS.

By default, PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS does not return “item" or “canceled and deleted" charge offers and discount offers. Set PIN_FLD_FLAGS in the input flist to get item, canceled, and deleted charge offer and discount offer instances. For these records, the information about the charge offer and discount offer instances are retrieved from the **/purchased_product** and **/purchased_discount** purchase and cancellation event objects.

> **Note:**
>
> Item, canceled, and deleted charge offer and discount offer instances are not returned if the charge offer's or discount offer's event objects are not recorded or if they are purged. For example, if BRM is configured to not record events with no balance impacts, records for these events are not returned.

PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS gets the purchase, cycle, and usage validity periods for charge offers and discount offers from the account's **/purchased_product** and **/purchased_discount** objects. It returns this information in the purchase, cycle, and usage START_T and END_T fields located in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays in the output flist. If a charge offer or discount offer is set to start on first usage and its validity period has not yet been initialized, additional information about the purchase, cycle, and usage start and end times is returned in START_DETAILS and END_DETAILS fields.

Getting a List of Packages and Bundles That an Account Owns

To get a list of the packages that an account owns, use PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS. This opcode retrieves a list of packages, bundles, or both that an account owns.

> **Note:**
>
> The opcode returns packages or bundles that an account *owns*. It does not return a package if the package has only optional bundles. The package must have at least one purchased bundle.

PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS takes an account POID from the input flist and returns a list of all packages, including balance group information and bundles the account owns.

If the account contains optional bundles that it does not own, those bundles are returned with the list as eligible to purchase (PIN_FLD_BOOLEAN value of **0**).

PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS can perform additional filtering of packages or bundles before they are returned. For example, add filtering logic to this opcode to return only inactive optional bundles.

Packages and bundles that were closed during a transition are not added to the output flist.

> **Note:**
>
> An account owning more than one instance of a package is not supported. In this case, PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS returns bundles for one of the packages selected at random.

PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS returns bundles in a **PIN_FLD_DEALS** array, even if it found them listed in the account in the older **PIN_FLD_DEAL_OBJ** field.

PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS is not called by any opcode.

Reading Data for All Valid Purchased Charge Offers and Discount Offers

To get the purchased charge offers and discount offers data for accounts and services, use PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS.

The value of the PIN_FLD_SCOPE_OBJ field in the input flist restricts the results to an account, bill unit (**/billinfo** object), or service. This field is required and can specify one of these values:

- An **/account** object: The opcode gets all the charge offers and discount offers for the account as well as its services.
- A **/billinfo** object: The opcode gets the purchased charge offers and discount offers that contribute to the given bill unit.
- A **/service** object: The opcode gets the purchased charge offers and discount offers that belong to the given service.

The opcode logic performs a search based on the scope object passed in the input flist. Most of the other parameters are filters that work on a particular scope. The input flist would look like this:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 618010 0
0 PIN_FLD_SCOPE_OBJ      POID [0] 0.0.0.1 /service/ip 615706 3
0 PIN_FLD_STATUS_FLAGS   INT [0] 3
0 PIN_FLD_VALIDITY_FLAGS INT [0] 3
0 PIN_FLD_INCLUSION_FLAGSINT [0] 2
0 PIN_FLD_OVERRIDE_FLAGS INT [0] 2
0 PIN_FLD_END_T          TSTAMP [0] (1154415600) Tue Aug  1 00:00:00 2006
0 PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 615702 3
0 PIN_FLD_OVERRIDDEN_OBJ  POID [0] 0.0.0.1 /purchased_product 324706 0
0 PIN_FLD_PACKAGE_ID INT [0] 23
0 PIN_FLD_PRODUCTS       ARRAY [*]     NULL array ptr
0 PIN_FLD_DISCOUNTS      ARRAY [*]     NULL array ptr
```

The opcode can accept filters to limit the search further. The input for the opcode contains parameters to retrieve the correct set of offerings:

- To retrieve only charge offers or only discount offers, use the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays.
- To retrieve only specific fields from the **/purchased_product** objects, specify the fields in the PRODUCTS and DISCOUNTS arrays.
- To retrieve charge offers and discount offers with a specific status, pass the PIN_FLD_STATUS_FLAGS field and specify one of these values:
  - PIN_SUBS_FLG_OFFERING_STATUS_ACTIVE: This flag retrieves only active offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_INACTIVE: This flag retrieves only inactive offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_CLOSED: This flag retrieves only closed offerings.

  > **Note:**
  >
  > Use of multiple values implies the target object should satisfy any of the above.

- To retrieve charge offers or discount offers valid as of a given time, pass the PIN_FLD_END_T field in the input flist. You can base the end time on only the purchase, cycle, or usage period by including the PIN_FLD_VALIDITY_FLAGS field in the input flist with one or more of these values.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_CYCLE: This flag tells the opcode to compare the END_T value passed to the CYCLE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_PURCHASE: This flag tells the opcode to compare the END_T value passed with PURCHASE_END_T.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_USAGE: This flag tells the opcode to compare the END_T value passed with USAGE_END_T.

  > **Note:**
  >
  > Use of multiple flags implies the target object must satisfy all of the above conditions.

- To retrieve all eligible offerings, which include account and subscription offerings, pass the PIN_FLD_INCLUSION_FLAGS field on the input flist with one or more of these values:
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_PRODS: This flag retrieves all eligible charge offers.
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_DISCS: This flag retrieves all eligible discount offers.

  > **Note:**
  >
  > When this field is missing, only eligible offerings from a given scope are returned.

- To retrieve only account charge offers and discount offers, pass the PIN_FLD_OVERRIDE_FLAGS field set to PIN_SUBS_FLG_OFFERING_ACCT_LEVEL_ONLY. This flag is valid for **/account** objects only.
- To retrieve charge offers and discount offers for only a single package, pass the PIN_FLD_PACKAGE_ID field set to the package ID. This field can be used with any scope.
- To retrieve charge offers and discount offers for a specific bundle, pass the PIN_FLD_DEAL_OBJ field set to the **/deal** object's POID.

PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS returns all purchased charge offers and discount offers, including canceled charge offers and discount offers, within the parameters used.

Finding Events Associated with Bundles, Charge Offers, Discount Offers, and Services

Use PCM_OP_SUBSCRIPTION_GET_HISTORY to retrieve the event history for a bundle, charge offer, discount offer, or service instance associated with an account.

This opcode retrieves events for a bundle, charge offer, discount offer, or service instance in a specified date range. Which events are returned depends on the category specified in the input flist:

- If a *bundle* is specified, a history of all events associated with the specific bundle instance is returned for the account. The bundle is identified by the PIN_FLD_PACKAGE_ID field in the input flist.
- If a *charge offer* is specified, a history of all events associated with the specific charge offer instance is returned for the account. The charge offer instance is identified by the PIN_FLD_OFFERING_OBJ field in the input flist.
- If a *discount offer* is specified, a history of all events associated with the specific discount offer instance is returned for the account. The discount offer instance is identified by the PIN_FLD_OFFERING_OBJ field in the input flist.
- If a *service* is specified, all events associated with that service object are returned for the account.

If you do not specify a date range, PCM_OP_SUBSCRIPTION_GET_HISTORY uses the account's creation time.

If a RESULTS array is passed in the input flist, PCM_OP_SUBSCRIPTION_GET_HISTORY returns only fields from the events that are passed in the RESULTS array.

Getting a Life-Cycle State

By default, after a balance is adjusted for a service that uses a custom life cycle, PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE triggers service state changes and then updates state expiration dates as follows:

1. Checks the **SubscriberLifeCycle** business parameter associated with a service's bill unit (**/billinfo** object):
   - If the parameter is set to **disabled**, returns the flow to the calling opcode.
   - If the parameter is set to **enabled**, continues the triggering process.

2. Gets the current life cycle state of the service from the PIN_FLD_LIFECYCLE_STATE field in the **/service** object.

   If the service POID is not in the opcode's input flist, calls PCM_OP_SEARCH to find all the services associated with the balance group and the **/account** object. If any of the retrieved services are in the Recharge Only or Credit Expired state, changes their state to Active.

3. Calls PCM_OP_BAL_GET_BALANCES to get the sum of the balances in the service balance group.

4. Calls PCM_OP_CUST_UPDATE_SERVICES to do the following:

   - If the service state is Active and the balance group has reached its credit limit, change the state to Recharge Only.
   - If the service state is Recharge Only and the balance is replenished, change the state to Active.
   - If the service state is Credit Expired and the balance is replenished, change the state to Active and update the PIN_FLD_SERVICE_STATE_EXPIRATION_T value in the **/service** object as follows:
     - If a voucher is applied to the balance, compare the voucher's validity period with the Active state's expiration period (PIN_FLD_SERVICE_STATE_EXPIRATION_T field in the **/config/lifecycle_states** object associated with the service), and update the expiration time based on the greater of the two periods.
     - If a voucher is not applied, update the expiration time based on the Active state's expiration period.
   - If the service state is Preactive or Active and a voucher is used to increase the balance, compare the voucher's validity period with the current state's expiration period, and update the expiration time based on the greater of the two periods.

   > **Note:**
   >
   > Oracle recommends that top-ups not be performed in the Preactive state.

By default, PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE supports the sample prepaid service life cycle. You can customize it to support other custom service life cycles.

PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE is called by PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS.

Getting Information about Price Tags

Use PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS to get information about price tag configuration objects.

Price tag configuration objects, stored in **/config/price_tags** objects, contain information about the resources, units, and services price tags can be used with, as well as values that can be used at purchase time.

You can get all price tags, or just price tags associated with specific resources, services, or products by including different fields in the input flist. At minimum, you must include the POID of a price tag configuration object. You can use a combination of the following additional fields:

- PIN_FLD_PRODUCT_OBJ
- PIN_FLD_NAME

- PIN_FLD_RESOURCE_ID
- PIN_FLD_RESOURCE_UNIT
- PIN_FLD_PERMITTED

You can use the information returned to determine which price tags are valid to add to products and discounts at design time, and which values are valid to use at purchase time.

The following examples show input and output flists for PCM_OP_SUBSCRIPTION_GET_PRICE_TAGS using different combinations of fields.

Get all price tags

This input flist requests all price tags under a specific price tag configuration object.

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /config/price_tags 10001 0
```

The response shows three price tags.

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_PRICE_TAGS  ARRAY [0]
1         PIN_FLD_NAME           STR [0] "PRICE1"
1         PIN_FLD_DESCRIPTION    STR [0] "Dollar Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"ANY"
1         PIN_FLD_CONSTRAINTS    STR [0] ""
1         PIN_FLD_RESOURCE_ID    INT [0] 840
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/email"
0 PIN_FLD_PRICE_TAGS  ARRAY [1]
1         PIN_FLD_NAME           STR [0] "PRICE2"
1         PIN_FLD_DESCRIPTION    STR [0] "Dollar Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"LIST"
1         PIN_FLD_CONSTRAINTS    STR [0] "10;20;50;100;200"
1         PIN_FLD_RESOURCE_ID    INT [0] 840
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/ip; /service/email"
0 PIN_FLD_PRICE_TAGS  ARRAY [2]
1         PIN_FLD_NAME           STR [0] "PRICE3"
1         PIN_FLD_DESCRIPTION    STR [0] "Euro Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"RANGE"
1         PIN_FLD_CONSTRAINTS    STR [0] "10;200"
1         PIN_FLD_RESOURCE_ID    INT [0] 978
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/email"
```

Get price tags for a resource

This input flist requests price tags under a specific price tag configuration object for a specific resource.

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_RESOURCE_ID       INT [0] 840
```

The response shows two price tags.

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_PRICE_TAGS  ARRAY [0]
1         PIN_FLD_NAME           STR [0] "PRICE1"
1         PIN_FLD_DESCRIPTION    STR [0] "Dollar Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"ANY"
1         PIN_FLD_CONSTRAINTS    STR [0] ""
1         PIN_FLD_RESOURCE_ID    INT [0] 840
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/email"
0 PIN_FLD_PRICE_TAGS  ARRAY [1]
1         PIN_FLD_NAME           STR [0] "PRICE2"
1         PIN_FLD_DESCRIPTION    STR [0] "Dollar Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"LIST"
1         PIN_FLD_CONSTRAINTS    STR [0] "10;20;50;100;200"
1         PIN_FLD_RESOURCE_ID    INT [0] 840
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/ip; /service/email"
```

Get price tags for a service

This input flist requests price tags under a specific price tag configuration object for a specific resource and service.

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_RESOURCE_ID       INT [0] 840
0 PIN_FLD_PERMITTED         STR [0] "/service/ip"
```

The response shows one price tag.

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_PRICE_TAGS  ARRAY [0]
1         PIN_FLD_NAME           STR [0] "PRICE2"
1         PIN_FLD_DESCRIPTION    STR [0] "Dollar Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"LIST"
1         PIN_FLD_CONSTRAINTS    STR [0] "10;20;50;100;200"
1         PIN_FLD_RESOURCE_ID    INT [0] 840
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/ip; /service/email"
```

Get price tags for a product

This example shows the input flist to get price tags under a specific price tag configuration object for a specific resource and service.

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_PRODUCT_OBJ      POID [0] 0.0.0.1 /product 786412 0
```

The response shows one price tag.

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /config/price_tags 10001 0
0 PIN_FLD_PRICE_TAGS  ARRAY [0]
1         PIN_FLD_NAME           STR [0] "PRICE3"
1         PIN_FLD_DESCRIPTION    STR [0] "Euro Override"
1         PIN_FLD_RULE_TYPE      STR [0]
"RANGE"
1         PIN_FLD_CONSTRAINTS    STR [0] "10;200"
1         PIN_FLD_RESOURCE_ID    INT [0] 978
1         PIN_FLD_RESOURCE_UNIT ENUM [0] 0
1         PIN_FLD_PERMITTED      STR [0] "/service/email"
```

Applying Promotions for Special Dates, Events, or Actions

Use the PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT opcode to apply promotions to accounts based on special dates, specific events, or specific actions. For example, you could grant 100 free minutes to customers on their birthday or every time they successfully top up their account balance.

This opcode is called by the following:

- The **pin_apply_promotion** utility to apply promotions to accounts based on special dates. See "pin_apply_promotion" in *BRM Setting Up Pipeline Pricing*.
- The event notification system when a triggering event occurs, such as when a customer purchases a new product. For information about the event notification system, see "Using Event Notification" in *BRM Developer's Guide*.

PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT performs these operations:

1. Checks the **/config/event_promo_tag_map** object to see whether the promotion tag (PIN_FLD_PROMO_TAG) is mapped to a promotion event (PIN_FLD_PROMO_EVENT_TYPE).
2. Checks the **/config/event_promo_map** object to see whether the event type (PIN_FLD_EVENT_TYPE) is mapped to a promotion event (PIN_FLD_PROMO_EVENT_TYPE).
3. Adds information about the promotion event to the input flist of the PCM_OP_ACT_USAGE opcode.
4. Calls the PCM_OP_SUBSCRIPTION_POL_PRE_PROMO_EVENT policy opcode to customize the input flist for the PCM_OP_ACT_USAGE opcode. By default, the policy opcode does nothing.
5. Calls the PCM_OP_ACT_USAGE opcode to rate the promotion event. To do so, it calls the following opcodes:
   - PCM_OP_RATE_POL_PRE_RATING. See "Modifying Rated Events".
   - PCM_OP_RATE_EVENT. See "Subscription Rating Opcodes".
   - PCM_OP_RATE_POL_POST_RATING. See "Modifying Rated Events".
6. Calls the PCM_OP_SUBSCRIPTION_POL_POST_PROMO_EVENT policy opcode to customize the output flist from the PCM_OP_ACT_USAGE opcode. By default, the policy opcode does nothing.

Customizing Special Date, Event, or Action Promotions

To customize how promotions are applied, use the following policy opcodes:

- PCM_OP_SUBSCRIPTION_POL_PRE_PROMO_EVENT: Customizes the input flist to PCM_OP_ACT_USAGE.

  By default, this opcode is an empty hook, but you can customize it to add or remove input flist fields.

- PCM_OP_SUBSCRIPTION_POL_POST_PROMO_EVENT: Customizes the output flist from PCM_OP_ACT_USAGE.

  By default, this opcode is an empty hook, but you can customize it to add or remove output flist fields.

Managing Promotions with Siebel CRM

To add, modify, inactivate, or cancel promotions, you must customize Siebel CRM to accept the following information and pass it to PCM_OP_SUBSCRIPTION_SET_BUNDLE:

- Promotion name
- Promotion description
- Charge offers and discount offers bundled with the promotion
- Promotion creation date
- Promotion validity dates
- Promotion status: active, inactive, or canceled

In an AIA system, you pass this information from Siebel CRM to BRM through the J2EE Connector Architecture (JCA) Resource Adapter, which exposes the BRM API through a JCA common client interface (CCI) as an Interaction. Data required for sending data to PCM_OP_SUBSCRIPTION_SET_BUNDLE is detailed in the Subscription Web service.

To implement the ability to display Siebel CRM promotions on invoices, add the following functionality to Siebel CRM by using PCM_OP_SUBSCRIPTION_SET_BUNDLE:

- Create **/purchased_bundle** objects. See "Adding a promotion to an account".
- Modify **/purchased_bundle** objects. See "Modifying an account's promotion".
- Inactivate **/purchased_bundle** objects. See "Inactivating an account's promotion".
- Cancel **/purchased_bundle** objects. See "Canceling an account's promotion".

Adding a promotion to an account

To add a promotion to a customer's account, pass in details about the promotion to PCM_OP_SUBSCRIPTION_SET_BUNDLE. You must also set the following fields in the opcode's input flist:

- In the PIN_FLD_BUNDLE_INFO array:
  - Set the PIN_FLD_POID field to a type-only POID for **/purchased_bundle**.
  - Set the PIN_FLD_STATUS field to **1** to set the promotion's status to **Active**.

> **Note:**
>
> If PIN_FLD_STATUS is not passed in, the promotion's status is set to active by default.

- In the PIN_FLD_OFFERINGS array, specify the charge offers and discount offers associated with the promotion. You must create a separate array for each charge offer and discount offer in the promotion. In the PIN_FLD_OFFERINGS array:
  - Set the PIN_FLD_POID field to the POID of the **/purchased_product** or **/purchased_discount** object.
  - Set the PIN_FLD_BUNDLE_OBJ field to NULL.
  - Set the PIN_FLD_BUNDLE_INFO array to the rec_id of the PIN_FLD_BUNDLE_INFO array at the 0th level of the flist. This specifies to associate the newly created **/purchased_bundle** object with the specified charge offer or discount offer during account creation.

Modifying an account's promotion

To modify an account's existing promotion, pass in the promotion details that changed to PCM_OP_SUBSCRIPTION_SET_BUNDLE:

- **To change a promotion's attributes**: In the PIN_FLD_BUNDLE_INFO array, set the PIN_FLD_POID field to the existing **/purchased_bundle** POID and then pass in only the fields that have changed.
- **To transfer a promotion from one account to another**: In the PIN_FLD_BUNDLE_INFO array, set the PIN_FLD_POID field to the existing **/purchased_bundle** POID and set the PIN_FLD_ACCOUNT_OBJ field to the new **/account** POID.
- **To add charge offers and discount offers to a promotion**: In the PIN_FLD_OFFERINGS array, set the PIN_FLD_POID field to the complete POID of the charge offer or discount offer being added to the promotion, and set the PIN_FLD_BUNDLE_OBJ field to the complete POID of the **/purchased_bundle** object.
- **To remove an existing charge offer or discount offer from a promotion**: In the PIN_FLD_OFFERINGS array, set the PIN_FLD_POID field to the complete POID of the charge offer or discount offer being removed from the promotion, and set the PIN_FLD_BUNDLE_OBJ field to NULL.

When a promotion is modified, the opcode does the following:

- Changes the specified **/purchased_bundle** object.
- Changes the PIN_FLD_PLAN_OBJ field of the **/purchased_product** or **/purchased_discount** object, if a charge offer or discount offer changed.
- Generates an **/event/billing/bundle/modify** object for auditing purposes, if a promotion's attributes changed.

Inactivating an account's promotion

When you inactivate an account's promotion, PCM_OP_SUBSCRIPTION_SET_BUNDLE sets the promotion's status to **Inactive**. It does not update the associated charge offer or discount offer's PIN_FLD_PLAN_OBJ field. Also, unlike the canceled status, a promotion's details can be modified after it has been inactivated.

To inactivate a promotion, pass in details about the promotion to PCM_OP_SUBSCRIPTION_SET_BUNDLE and set the PIN_FLD_STATUS field in the PIN_FLD_BUNDLE_INFO array to **2**.

Canceling an account's promotion

You can no longer make changes to an account's promotion after it has been canceled. When you cancel an account's promotion, PCM_OP_SUBSCRIPTION_SET_BUNDLE:

- Sets the promotion's validity end date to the current date.
- Sets the promotion's status to canceled.

Optionally, the opcode can disassociate the charge offers and discount offers from the promotion by changing the PIN_FLD_PLAN_OBJ field of the **/purchased_product** object and the **/purchased_discount** object to NULL.

To cancel an account's promotion, call PCM_OP_SUBSCRIPTION_SET_BUNDLE and set the **PIN_FLD_STATUS** field in the **PIN_FLD_BUNDLE_INFO** array to **3**.

To configure the opcode to disassociate the charge offers and discount offers from the promotion, perform one of the following tasks:

- Set the PIN_FLD_FLAGS field to **1**. When set, the opcode searches for and disassociates the account's charge offers and discount offers from the promotion. This option decreases processing performance.
- In the PIN_FLD_OFFERINGS array, set the PIN_FLD_POID field to the complete POID of the charge offer or discount offer being canceled, and set the PIN_FLD_BUNDLE_OBJ field to NULL. When set, the opcode disassociates the specified charge offers and discount offers from the promotion.

> **Note:**
>
> To improve processing performance, pass in all charge offers and discount offers associated with the canceled promotion.

Managing Provisioning

See the following topics:

- Customizing Provisioning When a Charge Offer Is Purchased
- Getting a List of Provisioning Tags
- Customizing Provisioning When Canceling a Charge Offer

> **Note:**
>
> The BRM provisioning tag framework is the preferred method of customizing provisioning when a charge offer is purchased or canceled.

When defining a configuration for the provisioning tag framework, you specify one or more opcodes to perform an action, such as creating an ERA. You can specify that an opcode run when a charge offer or discount is purchased, canceled, or both. You can use an existing opcode or design a custom opcode.

If the provisioning tag is designed to create an ERA, you can specify that PCM_OP_SUBSCRIPTION_PROVISION_ERA run at both purchase and cancellation time. This opcode creates, modifies, or deletes a profile (**/profile** object). ERAs are stored in profiles.

PCM_OP_SUBSCRIPTION_PROVISION_ERA creates, modifies, or deletes **/profile** objects. When specified in a **/config/provisioning_tag** object, this opcode runs when a charge offer or discount offer containing the provisioning tag is purchased or canceled. Profiles can store extended rating attributes (ERAs) and other custom attributes.

PCM_OP_SUBSCRIPTION_PROVISION_ERA calls PCM_OP_CUST_CREATE_PROFILE, PCM_OP_CUST_MODIFY_PROFILE, or PCM_OP_CUST_DELETE_PROFILE, depending on the action it must take.

When creating a profile, PCM_OP_SUBSCRIPTION_PROVISION_ERA creates a **/profile/acct_extrating** object for an account profile and a **/profile/serv_extrating** object for a service profile.

Customizing Provisioning When a Charge Offer Is Purchased

To set fields in a **/service** object at the time of purchase, use PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING.

This opcode is called by PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT when a charge offer is purchased.

> **Note:**
>
> Do not call PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING directly.

- If a **/service** object is associated with a provisioning tag in the input flist, provisioning is invoked.
- If provisioning is invoked, it checks the tag table for the corresponding provisioning tag and runs the corresponding function call that modifies the appropriate fields in the **/service** object.

You can customize the provisioning functionality as follows:

- Change the services, tags, and functions in the **provisioning_tags** array.
- Add to or modify the provisioning subfunctions in the **fm_subscription_pol_provisioning.c** file.
- Use PDC or Pricing Center to modify charge offers associated with services by including the applicable provisioning tag.

The BRM provisioning tag framework is the preferred method of customizing provisioning when a charge offer is purchased.

Getting a List of Provisioning Tags

To customize charge offer provisioning, use PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS. This opcode retrieves data for provisioning from the **provisioning_tags** array.

This opcode is called by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT when a charge offer is canceled.

> **Note:**
>
> Do not call PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS directly.

Each service in the **provisioning_tags** array is associated with a provisioning tag and a function. The provisioning tag specifies the type of service, and the function determines what is to be done with that service.

You can customize the provisioning functionality as follows:

- Change the services, tags, and functions in the **provisioning_tags** array.
- Add to or modify the provisioning subfunctions in the **fm_subscription_pol_provisioning.c** file.
- Use PDC or Pricing Center to modify charge offers associated with services by including the applicable provisioning tag.

Customizing Provisioning When Canceling a Charge Offer

To customize provisioning when canceling a charge offer, use PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING. This opcode clears fields in a **/service** object at the time of cancellation.

The BRM provisioning tag framework is the preferred method of customizing provisioning when a charge offer is canceled.

PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING is called by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT when a charge offer is canceled.

> **Note:**
>
> Do not call PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING directly.

- If a **/service** object is associated with a provisioning tag in the input flist, provisioning is invoked.
- If provisioning is invoked, it checks the tag table for the corresponding provisioning tag and runs the corresponding function call that clears the appropriate fields in the **/service** object.

Managing GSM Service Provisioning

Use the following opcodes to manage provisioning:

- To publish a service order, use PCM_OP_PROV_PUBLISH_SVC_ORDER. This opcode sends an **/event/provisioning/service_order/***eventtype* event to the Provisioning Data Manager (DM), where *eventtype* is the type of event.

  PCM_OP_PROV_PUBLISH_SVC_ORDER takes as input the POID of the service order event and a substruct containing the service order information that should be sent to the Provisioning DM.

- To update a service order, use PCM_OP_PROV_UPDATE_SVC_ORDER. This opcode updates the status of an **/event/provisioning/service_order/*** event.

  The **/event/provisioning/service_order/*** event stores the service order and information such as the status, service order type, and actions required.

  The PCM_OP_PROV_UPDATE_SVC_ORDER input flist includes the complete response from the provisioning applications, including the POID of the service order, the service order status, and other service order information. Based on the type of the service order, you can modify or validate the response flist.

  When a response is received from a provisioning platform, PCM_OP_PROV_UPDATE_SVC_ORDER uses information in the input flist to update the status of an **/event/provisioning/service_order/*** event.

- To validate and modify parameters for updating service orders, use PCM_OP_PROV_POL_UPDATE_SVC_ORDER.

  This opcode is called by PCM_OP_PROV_UPDATE_SVC_ORDER when a response is received from a provisioning system.

  By default, PCM_OP_PROV_POL_UPDATE_SVC_ORDER validates and transforms a global system for mobile communications (GSM) service order. When a response is received from a provisioning system, this opcode maps the response parameters to corresponding fields in the **/event/provisioning/service_order/telco/gsm** object.

  The input flist to PCM_OP_PROV_POL_UPDATE_SVC_ORDER includes the complete response from the provisioning applications. Based on the type of the service order, you can modify or validate the response flist.

  By default, PCM_OP_PROV_POL_UPDATE_SVC_ORDER returns the input flist without any change, except in the case of GSM services where the input flist is validated and the format is modified to map the fields in the **event/provisioning/service_order/telco/gsm** object.

Validating Product Specification Attributes for Pricing Components

To validate product specification attributes defined on pricing components, use PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING.

By default, this policy opcode is an empty hook, but you can customize it to validate product specification attributes.

PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING is called by the following opcodes:

- PCM_OP_SUBSCRIPTION_PURCHASE_DEAL
- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT
- PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT
- PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE
- PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY

See *BRM Developer's Guide* for information about customizing opcodes.

23

Tax Calculation Opcode Workflows

Learn about the Oracle Communications Billing and Revenue Management (BRM) tax calculation opcode workflows.

Topics in this document:

- Opcodes Described in This Chapter
- Calculating Taxes during Billing
- Modifying Tax Data Before Calculating Taxes
- Modifying Tax Data After Calculating Taxes
- Overriding Customer Tax Locale for Purchases
- Using Custom Tax Rates
- Using Geocodes to Calculate Taxes
- Adding Tax Information to Accounts
- Retrieving Tax Calculation Data
- Retrieving Additional Tax Data from Vertex Communications Tax Q Series

Opcodes Described in This Chapter

Table 23-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 23-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_BILL_CYCLE_TAX	Calculating Taxes during Billing
PCM_OP_BILL_TAX_EVENT	Retrieving Tax Supplier Data
PCM_OP_CUST_POL_TAX_CALC	Using Custom Tax Rates
PCM_OP_CUST_POL_TAX_INIT	Using Custom Tax Rates
PCM_OP_CUST_POL_VALID_NAMEINFO	Using Geocodes to Calculate Taxes
PCM_OP_CUST_POL_VALID_TAXINFO	Validating Tax Information
PCM_OP_CUST_SET_TAXINFO	Adding Tax Information to Accounts

Table 23-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_RATE_EVENT	Calculating Taxes during Billing Retrieving Tax Location Data
PCM_OP_RATE_POL_GET_TAX_SUPPLIER	Retrieving a List of Tax Suppliers
PCM_OP_RATE_POL_GET_TAXCODE	Retrieving a List of Tax Codes
PCM_OP_RATE_POL_MAP_TAX_SUPPLIER	Retrieving Tax Supplier Data
PCM_OP_RATE_POL_POST_TAX	Modifying Tax Data After Calculating Taxes Requesting Additional Data from Vertex Communications Tax Q Series Retrieving Additional Tax Data Storing Vertex Tax Data in Your Custom Storable Class
PCM_OP_RATE_POL_PRE_TAX	Modifying Tax Data Before Calculating Taxes Using Geocodes to Calculate Taxes Requesting Additional Data from Vertex Communications Tax Q Series
PCM_OP_RATE_POL_TAX_LOC	Calculating Taxes during Billing Retrieving Tax Location Data
PCM_OP_RATE_TAX_CALC	How PCM_OP_RATE_TAX_CALC Calculates Taxes Calculating Taxes during Billing Modifying Tax Data Before Calculating Taxes Validating Tax Information

Calculating Taxes during Billing

Taxes are calculated during rating and during billing. During rating, taxes are calculated by ECE. You can customize tax calculation in rating by using ECE extensions.

To calculate taxes during billing:

1. For each rated event, PCM_OP_RATE_EVENT reads the PIN_FLD_RATES_USED array in the input flist to determine what pricing to apply. This pricing information determines how the taxable amount is treated.
2. PCM_OP_RATE_EVENT calls PCM_OP_RATE_POL_TAX_LOC to determine whether custom processing or default processing is used to determine the tax-related locales.
   - If default processing is used, PCM_OP_RATE_EVENT retrieves the locales from the database.
   - If custom processing is used, control is handed to PCM_OP_RATE_POL_TAX_LOC, which determines which locales to use.

   The locales are used during the tax calculation process to determine jurisdiction. BRM stores up to four tax locales for the current account, session, and event.

   See "Retrieving Tax Location Data" for more information on the tax locales and how to customize their retrieval.
3. PCM_OP_RATE_EVENT also checks the PIN_FLD_EXEMPTIONS array in **/account** objects to determine whether all or part of the purchase amount is exempt from taxes.

4. PCM_OP_RATE_EVENT returns a revised set of objects for the event. This can include PIN_FLD_TAXES and PIN_FLD_EXEMPTIONS arrays to indicate tax amounts that were calculated immediately by PCM_OP_RATE_TAX_CALC.
5. The event is taxable, PCM_OP_RATE_EVENT calls PCM_OP_BILL_CYCLE_TAX. If the event is nontaxable, PCM_OP_RATE_EVENT does nothing.
6. If a transaction is not already open, PCM_OP_BILL_CYCLE_TAX opens one.

   If PCM_OPFLG_CALC_ONLY is set, the transaction is opened with the flag PCM_TRANS_OPEN_READONLY.
7. PCM_OP_BILL_CYCLE_TAX calls PCM_OP_RATE_TAX_CALC to perform the actual tax calculation. See "How PCM_OP_RATE_TAX_CALC Calculates Taxes" for information.
8. PCM_OP_BILL_CYCLE_TAX receives the tax data from PCM_OP_RATE_TAX_CALC in the PIN_FLD_TAXES array.
9. PCM_OP_BILL_CYCLE_TAX does one of the following:
   - Creates an **/event/billing/cycle/tax** object with the taxable amount if there were any deferred taxes.
   - If PCM_OP_BILL_CYCLE_TAX was called with the PCM_OPFLG_CALC_ONLY flag set, it still performs the tax calculation but does not create the **/event/billing/cycle/tax** object.

     Instead, the PCM_OP_BILL_CYCLE_TAX output flist contains the PIN_FLD_RESULTS array, an flist with the information that would have been used to create the **/event/billing/cycle/tax** object.
10. PCM_OP_BILL_CYCLE_TAX sets the **/event/billing/cycle/tax** object status to inactive.
11. PCM_OP_BILL_CYCLE_TAX commits the transaction if one was opened.

How PCM_OP_RATE_TAX_CALC Calculates Taxes

PCM_OP_RATE_TAX_CALC determines the taxation Data Manager (DM) or custom taxation method to be used for an event. It sends the input flist to the DM or the custom tax calculation method for calculation. It receives the information it needs to calculate taxes from the following input fields:

- The tax code of the BRM charge offer mapped to a charge offer code that is recognized by the taxation software. The tax code also identifies the taxation DM or the custom tax calculation method to be used.
- The purchase amount to be taxed from the PIN_FLD_TAXES array. Each element of the PIN_FLD_TAXES array represents a separate tax calculation method.

> **Note:**
>
> If the tax calculation software or custom policy is not specified in the tax code, taxes are not calculated.

- The locations that are involved in the transaction to be taxed. This includes ship-from, ship-to, order origin, and order accept addresses for sales and use taxation.

> **Note:**
>
> For telecommunications taxation, the address fields contain additional information (NPA-NXX or geocodes) for origin, termination, and charge-to numbers.

- The date and time of the transaction from the PIN_FLD_START_T and PIN_FLD_END_T fields. PIN_FLD_START_T is used to compute PIN_FLD_ELAPSED_TIME or duration (that is, the difference between the end time and start time).
- Tax exemption information, including exemption information based on tax jurisdiction, from the PIN_FLD_EXEMPTIONS array.
- The tax supplier ID from the PIN_FLD_TAX_SUPPLIER field.
- Other optional values such as the account's bill object in the PIN_FLD_BILL_OBJ field and telecommunications values such as incorporated, residence, and regulated flags.

PCM_OP_RATE_TAX_CALC checks for the following errors:

- Missing or nonvalid PIN_FLD_TAX_CODE field.
- Nonvalid PIN_FLD_TAXPKG_TYPE field, which is derived from PIN_FLD_TAX_CODE field and the **taxcodes_map** file.
- Missing or nonvalid tax database entry.

PCM_OP_RATE_TAX_CALC returns an output flist with a revised PIN_FLD_TAXES array containing the tax data that the tax calculation software or the custom tax calculation method returns. The calculated taxes are sorted by jurisdiction.

If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_RATE_TAX_CALC performs the tax calculation and returns the tax rate and amount. The tax data is not written to the database, however.

How PCM_OP_RATE_TAX_CALC Validates Addresses and VAT Certificates

You can call PCM_OP_RATE_TAX_CALC in jurisdiction-check-only mode during account creation to perform address or VAT certificate number validation. You specify jurisdiction-check-only mode by using the PIN_FLD_COMMAND input field. The following modes are supported:

- If PIN_FLD_COMMAND is set to **1**, it sends an address to the taxation DM for verification and additionally returns the geocode and county information in the PIN_FLD_NAMEINFO array if the address was successfully validated.
- If PIN_FLD_COMMAND is set to **2**, it sends a VAT certificate number to the tax DM for validation.

> **Note:**
>
> The VAT certificate number validation is supported only for a custom tax calculation implementation. It is not supported by the default BRM tax calculation functions.

In jurisdiction-check-only mode, PCM_OP_RATE_TAX_CALC returns **1** if the jurisdiction is valid and **0** if it is not.

Policy Opcodes Called by PCM_OP_RATE_TAX_CALC

PCM_OP_RATE_TAX_CALC calls the following policy opcodes that you can customize to modify the tax data or to use custom tax calculation:

- PCM_OP_RATE_POL_PRE_TAX, to modify any tax data in the input flist that is passed to the taxation DM.

  See "Modifying Tax Data Before Calculating Taxes" for more information.
- PCM_OP_RATE_POL_POST_TAX, to modify any tax data in the output flist that is returned from the taxation DM.

  See "Modifying Tax Data After Calculating Taxes" for more information.
- PCM_OP_CUST_POL_TAX_CALC, to use custom tax calculation methods instead of using an external tax calculation software.

  See "Using Custom Tax Rates" for more information.

Modifying Tax Data Before Calculating Taxes

Use PCM_OP_RATE_POL_PRE_TAX to modify data before you send the data to the taxation DM for calculating taxes.

This opcode is called by PCM_OP_RATE_TAX_CALC before determining the tax package to use for tax calculation.

By default, PCM_OP_RATE_POL_PRE_TAX returns the input flist as the output flist. You can customize the opcode to perform these functions:

- Identify the geocode for an address and provide it to the taxation DM.
- Add a tax exemption to the input flist before sending the flist to the taxation DM.
- Change the tax exemption from a percentage to a set amount. For example, deduct x amount from the taxable amount because the first x amount is exempt from taxes.

In addition, the opcode source file contains commented code for enhancements available to Vertex Communications Tax Q Series users.

Customizing Vertex Communications Tax Q Series to Override ZIP Codes

For Vertex Communications Tax Q Series, the PCM_OP_RATE_POL_PRE_TAX policy source file contains commented code to instruct the Vertex Communications Tax Q Series taxation software to override the ZIP code. This enables the Vertex DM to charge taxes for nontelephony events by using Vertex Communications Tax Q Series.

To customize Vertex Communications Tax Q Series to override ZIP codes:

1. Open the PCM_OP_RATE_POL_PRE_TAX policy source file with a text editor.
2. Uncomment the code.
3. Define the OVERRIDE_BY_ZIP symbol.
4. Save and close the file.
5. Recompile the opcode.

Customizing Vertex Communications Tax Q Series to Provide Custom Input Tax Data

For Communications Tax Q Series, you can modify PCM_OP_RATE_POL_PRE_TAX source code to use a customized input flist to the taxation DM for tax data. The taxation DM checks the input flist. If it includes the PIN_FLD_STATUS_FLAGS field with the value **1**, the DM sets all custom fields included in the flist and then calls the tax package to calculate taxes.

> **Note:**
>
> When you create the customized flist, make sure that you express the field values on the basis of Vertex field data types. Add the correct data type and value for each Vertex attribute that you want to customize. To see your customized flist, you can use PIN_ERR_LOG_FLIST in debug mode.

To customize PCM_OP_RATE_POL_PRE_TAX:

1. Open *BRM_SDK_home***/source/sys/fm_rate_pol/fm_rate_pol_pre_tax.c** file.
2. Add the code from "Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX" to the **fm_rate_pol_pre_tax()** function after the OVERRIDE_BY_ZIP section.
3. Add optional fields to customize the tax calculation process (the source code includes a comment about how to add new fields).
4. Define the USE_VERTEX_Q_SERIES symbol.
5. Add **stda.h** and **ctqa.h** to the Include section of the file.
6. Set the value of **customized_flag** to **1**, if it is not already set to that value.
7. Save and close the file.
8. Copy the **stda.h** and **ctqa.h** files from the *Vertex_home***/inc** directory to the *BRM_SDK_home***/source/sys/fm_rate_pol** directory.
9. To the **Makefile**, add CFLAGS += -DPORT_UNIXANSI.
10. Recompile the opcode.

Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX

```
#ifdef USE_VERTEX_Q_SERIES
/*************************************************************************
      /*****************************************************************
      * FOLLOWING MANDATORY FIELDS SHOULD BE FILLED WITH CORRECT VALUES.
      * YOU CAN CUSTOMIZE THE MANDATORY AND OPTIONAL FIELDS.
      * HERE IS THE LIST OF MAPPING FIELDS.
      * VERTEX DATATYPE => MAPPING FIELD              =>PORTAL DATATYPE
      *
      * eCtqString    =>PIN_FLD_VERTEX_CTQ_STR        =>PIN_FLDT_STR
      * eCtqInt       =>PIN_FLD_VERTEX_CTQ_INT        =>PIN_FLDT_INT
      * eCtqFloat     =>PIN_FLD_VERTEX_CTQ_FLOAT      =>PIN_FLDT_DECIMAL
      * eCtqDouble    =>PIN_FLD_VERTEX_CTQ_DOUBLE     =>PIN_FLDT_DECIMAL
      * eCtqLong      =>PIN_FLD_VERTEX_CTQ_LONG       =>PIN_FLDT_DECIMAL
      * eCtqDate      =>PIN_FLD_VERTEX_CTQ_DATE       =>PIN_FLDT_TSTAMP
      * eCtqTime      =>PIN_FLD_VERTEX_CTQ_TIME       =>PIN_FLDT_TSTAMP
```

```
* eCtqTimestamp =>PIN_FLD_VERTEX_CTQ_TIMESTAMP  =>PIN_FLDT_TSTAMP
* eCtqBool      =>PIN_FLD_VERTEX_CTQ_BOOL       =>PIN_FLD_INT
* eCtqChar      =>PIN_FLD_VERTEX_CTQ_CHAR       =>PIN_FLDT_STR
* eCtqEnum      =>PIN_FLD_VERTEX_CTQ_ENUM       =>PIN_FLDT_ENUM
*
* YOU CAN ADD ONE FIELD LIKE THE FOLLOWING FLIST FORMAT.
0 PIN_FLD_FIELD_NAMES   ARRAY[] allocated 20, used 2
        1     PIN_FLD_TYPE             ENUM [0]
        1     PIN_FLD_VERTEX_CTQ_STR    STR [0]
* USE PROPER MAPPING FIELD TO SET VALUE OF VERTEX FIELD.
* IF YOU ADD SOME STRING FIELD(eCtqString),THEN SET VALUE FOR THAT FIELD
* USING PIN_FLD_VERTEX_CTQ_STR.FOR DETAILS,SEE THE ABOVE TABLES.
*****************************************************************/
pin_flist_t     *cust_flistp = NULL;
pin_flist_t     *field_name_flistp = NULL;
tCtqAttribType  attribute_type = eCtqHandle ;
time_t          now_t = 0;
pin_decimal_t   *value_decimal = NULL;
int32           value_int = 0;
char            temp_buf[255] = { '\0' };

int32           customized_flag = 1;/* SET TO 1 FOR CUSTOMIZATION */

cust_flistp = PIN_FLIST_CREATE(ebufp);

/*****************************************************************
 * IMPORTANT:SET customized_flag to 1 IF YOU WANT TO CUSTOMIZE THE
 * VERTEX FIELDS.
 *****************************************************************/

PIN_FLIST_FLD_SET(cust_flistp, PIN_FLD_STATUS_FLAGS,
        (void *)&customized_flag, ebufp);

/* FIELD:eCtqAttribCategoryCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribCategoryCode, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"04") ;
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribServiceCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribServiceCode, ebufp);

attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf,"01" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribOriginGeoCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribOriginGeoCode, ebufp);
```

```
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"390290740" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribTerminationGeoCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribTerminationGeoCode, ebufp);

attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf,"390296350") ;
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,(void *)temp_buf,
        ebufp);

/* FIELD:eCtqAttribInvoiceDate NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribInvoiceDate, ebufp);
attribute_type=eCtqDate;
now_t = pin_virtual_time((time_t *)NULL);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DATE,&now_t,
        ebufp);

/* FIELD:eCtqAttribTaxableAmount NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribTaxableAmount, ebufp);
attribute_type = eCtqDouble;
value_decimal = pbo_decimal_from_str("100.00",ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DOUBLE,
        (void *)value_decimal,ebufp);


/*********************************************************************
 * YOU CAN ADD OPTIONAL FIELDS LIKE BELOW FLIST
 *
 * field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp,
        PIN_FLD_FIELD_NAMES,eCtqAttribBilledLines, ebufp);
 * attribute_type = eCtqInt;
 * PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
 * int value_int = 01
 * PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_INT,
        (void *)&value_int,ebufp);
 *********************************************************************/
 field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribBilledLines, ebufp);
 attribute_type = eCtqInt;
 value_int = 01;
 PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
```

```
         PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_INT,
                (void *)&value_int,ebufp);
        field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
                eCtqAttribSaleResaleCode, ebufp);
        attribute_type = eCtqString;
        temp_buf[0]='\0';
        strcpy(temp_buf ,"S" );
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
                (void *)&attribute_type,ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
                temp_buf,ebufp);

        field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
                eCtqAttribChargeToGeoCode, ebufp);
        attribute_type = eCtqString;
        temp_buf[0]='\0';
        strcpy(temp_buf ,"390296350" );
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
                (void *)&attribute_type,ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
                temp_buf,ebufp);

        field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
                eCtqAttribCallMinutes, ebufp);
        attribute_type = eCtqDouble;
        value_decimal = pbo_decimal_from_str("10.0",ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
                (void *)&attribute_type,ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DOUBLE,
                (void *)value_decimal,ebufp);

        field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
                eCtqAttribUserArea, ebufp);
        attribute_type = eCtqString;
        temp_buf[0]='\0';
        strcpy(temp_buf ,"Interstate/Toll");
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
                (void *)&attribute_type,ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,temp_buf,
                ebufp);

        field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
                eCtqAttribInvoiceNumber, ebufp);
        attribute_type = eCtqString;
        temp_buf[0]='\0';
        strcpy(temp_buf ,"DEMO-002");
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
                (void *)&attribute_type,ebufp);
        PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,temp_buf,
                ebufp);

        PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG,
                "op_rate_pol_pre_tax customized flist", cust_flistp);
        PIN_FLIST_CONCAT(ret_flistp,cust_flistp,ebufp);
        if (value_decimal) {
                pbo_decimal_destroy(&value_decimal);
        }

        PIN_FLIST_DESTROY_EX(&cust_flistp,NULL);
***********************************************************************/
#endif
```

Modifying Tax Data After Calculating Taxes

You can customize PCM_OP_RATE_POL_POST_TAX to modify any data in the output returned by the tax calculation software after tax calculation. For example, you can modify the opcode to perform these functions:

- Change how the CM logs any messages that are returned from the external taxation DMs.
- Add a surcharge to the tax amount in the output flist returned from the taxation DM, as shown in the following example:

```
void fm_rate_pol_post_tax(...)
{
/* loop through array of incoming taxes */
  while(t_flistp = PIN_FLIST_ELEM_GET_NEXT(in_flistp, PIN_FLD_TAXES,
      &elemid, 1, &cookie, ebufp)) != (pin_flist_t*)NULL) {
     /* add a surcharge to the SUBTOTAL array */
      cnt = PIN_FLIST_ELEM_COUNT(t_flist, PIN_FLD_SUBTOTAL, ebufp))
      s_flistp = PIN_FLIST_ELEM_ADD(t_flistp, PIN_FLD_SUBTOTAL, cnt++, ebufp);
...
     /* add the entries to the SUBTOTAL array */
      PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_TAX, (void*)&tax, ebufp);
      PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_SUBTYPE, (void*)&taxType, ebufp);
...
  }
}Error Handling
```

By default, PCM_OP_RATE_POL_POST_TAX returns the input flist as the output flist, without the PIN_FLD_MESSAGES array. It also logs any messages returned from the taxation DM as specified in the **tax_return_loglevel** entry of the CM **pin.conf** file.

PCM_OP_RATE_POL_POST_TAX default implementation handles errors as follows:

- Logs messages as specified in the **tax_return_loglevel** entry in the CM **pin.conf** file. The default is to log only warnings.
- If an error occurs while reading the **tax_return_loglevel** entry, sets the error buffer to INVALID_CONF.

Overriding Customer Tax Locale for Purchases

By default, BRM calculates the taxes your customers owe for purchases using their shipped-to address (stored in an ***/account*** object's PIN_FLD_NAMEINFO array). You can instead configure BRM to calculate the tax amount using a customer's real-time location when purchasing a bundle (or deal), buying a charge offer (or product), or topping up a prepaid account.

To do so, configure your custom client application to call the following opcodes:

- PCM_OP_SUBSCRIPTION_PURCHASE_DEAL
- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT
- PCM_OP_PYMT_TOPUP

You pass in the customer's real-time location to the opcode using the PIN_FLD_LOCALE input flist field or using zone maps.

Real-Time Location Using PIN_FLD_LOCALE

Set the PIN_FLD_LOCALE input flist field to the customer's current location in the following format:

```
"City;State;Zipcode;Country"
```

For example:

```
"Austin;TX;78741;US"
```

The following shows a sample PCM_OP_PYMT_TOPUP input flist for setting a customer's real-time location to Chicago:

```
0 PIN_FLD_PROGRAM_NAME                   STR [0] "testnap"
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 91058 8
0 PIN_FLD_LOCALE                         STR [0] "Chicago;IL;60634;US"
0 PIN_FLD_INHERITED_INFO           SUBSTRUCT [0] allocated 1, used 1
1 PIN_FLD_TOPUP_INFO                   ARRAY [0] allocated 4, used 4
2       PIN_FLD_BILLINFO_OBJ            POID [0] 0.0.0.1 /billinfo 94130 0
2       PIN_FLD_BAL_GRP_OBJ             POID [0] 0.0.0.1 /balance_group 93298
0
2       PIN_FLD_TOPUP_AMT            DECIMAL [0] 19
2         PIN_FLD_PAYINFO              ARRAY [0] allocated 4, used 4
3             PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account 91058 8
3             PIN_FLD_POID              POID [0] 0.0.0.1 /payinfo/cc -1 0
3             PIN_FLD_PAY_TYPE          ENUM [0] 10003
3             PIN_FLD_CC_INFO          ARRAY [0] allocated 9, used 9
4                 PIN_FLD_SECURITY_ID     STR [0] "ID"
4                 PIN_FLD_NAME            STR [0] "John Evans"
4                 PIN_FLD_DEBIT_EXP       STR [0] "Expiration"
4                 PIN_FLD_DEBIT_NUM       STR [0] "DebitNumber"
4                 PIN_FLD_COUNTRY         STR [0] "US"
4                 PIN_FLD_ZIP             STR [0] "94065"
4                 PIN_FLD_STATE           STR [0] "CA"
4                 PIN_FLD_CITY            STR [0] "Redwood City"
4                 PIN_FLD_ADDRESS         STR [0] "100 Oracle Pkwy"
```

Real-Time Location Using Zone Maps

Set the following opcode input flist fields:

- PIN_FLD_ZONEMAP_NAME: Set this to the name of the zone map to use.
- PIN_FLD_ZONEMAP_TARGET: Set this to the name of the end-leaf node in the zone map.

The following shows a sample PCM_OP_SUBSCRIPTION_PURCHASE_DEAL input flist for setting a customer's real-time location using zone maps:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 101267 7
0 PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.1 /service/ip 98643 6
0 PIN_FLD_PROGRAM_NAME       STR [0] "testnap"
0 PIN_FLD_ZONEMAP_TARGET     STR [0] "3"
0 PIN_FLD_ZONEMAP_NAME       STR [0] "TestZoneMap2"
```

```
0 PIN_FLD_DEAL_INFO    SUBSTRUCT [0] allocated 20, used 7
1    PIN_FLD_PRODUCTS      ARRAY [0] allocated 35, used 35
2       PIN_FLD_FLAGS        INT [0] 1
```

> **Note:**
>
> To use this method, you must have configured a zone map with the specified name and target.

Using Custom Tax Rates

If your business uses a simple tax calculation method, you may not need to use a tax calculation software package. You can use custom tax rates instead. In simple cases, you can implement taxation using built-in features that do not require programming.

If your business requires a more complex tax structure but does not require the full capabilities of one of the taxation packages, you can implement custom tax rates by modifying the two taxation policy opcodes:

- **PCM_OP_CUST_POL_TAX_INIT**: This policy opcode caches tax rate information from the **/config/taxcodes_map** object. Then, it calls either the PCM_OP_CUST_POL_TAX_CALC policy opcode (if no tax calculation software is configured) or the taxation DM (if tax calculation software is configured). PCM_OP_CUST_POL_TAX_INIT is not called by any other opcode.
- **PCM_OP_CUST_POL_TAX_CALC**: This policy opcode performs the actual tax calculation for all tax codes with a package code of **U** (user-defined). By default, PCM_OP_CUST_POL_TAX_CALC reads the custom tax rate from cache and uses this simple calculation method to calculate the taxes:
  tax = amount * rate

To use custom tax rates, do the following:

1. Define your tax rates as required for your BRM 15.0 release. See "Creating Tax Codes" in *BRM Calculating Taxes*.
2. Modify the source file for PCM_OP_CUST_POL_TAX_INIT (*BRM_SDK_home***/source/sys/fm_cust_pol/fm_cust_pol_tax_init.c**) to implement your customizations.
   a. To load tax code data in a different format, define a data structure in **fm_cust_pol_tax_init.c** for your custom tax rates.
   b. Perform any other customizations required in **fm_cust_pol_tax_init.c** to implement your custom tax features.
3. If necessary for your customizations, modify the source file for PCM_OP_CUST_POL_TAX_CALC (*BRM_SDK_home***/source/sys/fm_cust_pol_tax_calc.c**).
4. Compile **fm_cust_pol_tax_init.c** and **fm_cust_pol_tax_calc.c** by using the makefile in the *BRM_SDK_home***/source/sys/fm_cust_pol** directory.
5. Copy the compiled shared library files to *BRM_home***/lib**, replacing the existing versions.
6. Open the CM configuration file (**pin.conf**) and ensure that the following entries in the **fm_module** section are enabled (not commented out):

```
- cm fm_module BRM_home/lib/fm_cust.so fm_cust_config - pin
- cm fm_module BRM_home/lib/fm_cust_pol.so fm_cust_pol_config fm_cust_pol_tax_init pin
```

7. Stop and restart the CM.

Using Geocodes to Calculate Taxes

A geocode is a geographic code that is used to determine a tax jurisdiction. Different tax calculation software packages use different geocode systems and update them frequently. To use geocodes to calculate taxes, you must modify PCM_OP_RATE_POL_PRE_TAX to query the geocoder and obtain the geocode for an account during account creation. The geocoder maps the address or postal code of the account to a unique geocode.

If a geocode is available during tax calculation, the tax package uses it to calculate the tax. If a geocode is not available, the tax calculation software chooses the geocode with the highest tax rate from a list of geocodes it finds for that account address.

> **Note:**
>
> The tax calculation software itself cannot map an address to a unique geocode.

To use geocodes to calculate taxes:

1. Using Storable Class Editor, create a **/profile** storable class and add the PIN_FLD_GEOCODE field to it.

   If you have configured more than one tax calculation package, create an array that contains separate PIN_FLD_GEOCODE fields for the tax calculation package. The index values should represent the tax packages:

   - **0** specifies a custom tax package.
   - **2** specifies the WorldTax package.
   - **3** specifies the Sales Tax Q Series package.
   - **4** specifies the Communications Tax Q Series package.

2. Modify PCM_OP_CUST_POL_VALID_NAMEINFO to store the geocode in the new PIN_FLD_GEOCODE field of the **/profile** object if the address was validated successfully.
3. Implement PCM_OP_RATE_POL_PRE_TAX to perform the following functions:
   - Obtain the geocode from the **/profile** object.
   - Set the geocode in the PIN_FLD_SHIP_TO or PIN_FLD_SHIP_FROM field in the taxes array in the input flist to the taxation DM.

     For example, the opcode should change this field value:

     ```
     PIN_FLD_SHIP_TO STR [0] "Cupertino; CA; 95012; US; []"
     ```

     to include the geocode in the square brackets []:

     ```
     PIN_FLD_SHIP_TO STR [0] "Cupertino; CA; 95012; US; [050850860]"
     ```

4. Set the location mode to geocode by supplying **1** as the value for the PIN_FLD_LOCATION_MODE field in the PIN_FLD_TAXES array of the input flist.

When you set the location mode to geocode, the taxation DM passes the geocode instead of the address to the tax calculation module.

For example:

```
PIN_FLD_LOCATION_MODE  ENUM[0] 1
```

After calculating the taxes, the taxation DM returns the jurisdiction (address and geocode) where tax rates were applied in the PIN_FLD_TAXES array of the output flist:

```
PIN_FLD_NAME STR[0] "US: CA: Santa Clara: Cupertino: 95012 : [050850860]"
```

Adding Tax Information to Accounts

To add tax information to an account, use PCM_OP_CUST_SET_TAXINFO.

This opcode adds the following data:

- VAT certificate
- Tax exemptions
- Tax incorporation
- Tax residence

To do so, PCM_OP_CUST_SET_TAXINFO adds or updates values for the following fields in the **taxinfo** array of an **/account** object:

- PIN_FLD_VAT_CERT
- PIN_FLD_EXEMPTIONS
- PIN_FLD_INCORPORATED_FLAG
- PIN_FLD_RESIDENCE_FLAG

For example, when an account is created and a VAT certificate number or exemption information is provided, PCM_OP_CUST_SET_TAXINFO is called. PCM_OP_CUST_SET_TAXINFO performs these functions:

- Modifies existing data or adds information to the PIN_FLD_TAXINFO array of the **/account** object by using the field values provided in the input flist.
- If a VAT certificate number is not provided, replaces the value of PIN_FLD_VAT_CERT in the account object with an empty ("") string.
- If exemption information is not provided, deletes the PIN_FLD_EXEMPTIONS array.
- If the field values provided in the input flist contain the PIN_FLD_INCORPORATED_FLAG or PIN_FLD_RESIDENCE_FLAG flag, adds or updates these values in the account object.

Validating Tax Information

To customize how tax information is validated, use PCM_OP_CUST_POL_VALID_TAXINFO. This opcode validates the VAT certificate number to prevent nonvalid numbers, which cause failures in tax calculations.

By default, PCM_OP_CUST_POL_VALID_TAXINFO checks for the VAT certificate data and performs these functions:

- If the VAT certificate number provided is a string, zero in length, returns the validation result **PASS**.

- If the following **tax_valid** entry in the CM **pin.conf** file is not set or is set to **0**, returns the validation result **FAIL**.

```
#Enables or disables validation of the VAT certificate number.
#1 means Enable; 0, which is the default, means Disable.
- fm_cust_pol  tax_valid 0
```

You can customize PCM_OP_CUST_POL_VALID_TAXINFO to use a custom tax calculation software to validate the VAT certificate number. For example, instead of calling PCM_OP_RATE_TAX_CALC, you can query your custom VAT validation function by rewriting the function **do_vat_cert_validation** in PCM_OP_CUST_POL_VALID_TAXINFO source file.

Retrieving Tax Calculation Data

See the following topics:

- Retrieving a List of Tax Codes
- Retrieving a List of Tax Suppliers
- Retrieving Tax Supplier Data
- Retrieving Tax Location Data

Retrieving a List of Tax Codes

To retrieve a list of tax codes, use PCM_OP_RATE_POL_GET_TAXCODE. This opcode returns a list of all tax codes that were loaded from the **taxcodes.map** file and cached by the CM during initialization. For example, Pricing Center uses this opcode to display a list of taxcodes used to configure rate plans for taxation. You can customize this opcode to return additional cached tax code information.

This opcode is not called by any opcode.

PCM_OP_RATE_POL_GET_TAXCODE takes an account POID as input and performs these functions:

- Obtains the database number from the POID.
- Searches the CMs in-memory cache for tax codes.
- Returns an array of tax code names in the output flist.

PCM_OP_RATE_POL_GET_TAXCODE returns a list of the tax code names cached from the **taxcodes_map** file by the CM during initialization.

Retrieving a List of Tax Suppliers

To retrieve a list of tax suppliers, use PCM_OP_RATE_POL_GET_TAX_SUPPLIER. You can customize this opcode by modifying the fields on the output flist. You can specify which fields are validated by adding or removing them from the input flist.

PCM_OP_RATE_POL_GET_TAX_SUPPLIER takes an account POID as input and performs these functions:

- Obtains the database number from the POID.
- Performs a global search for tax suppliers.
- Returns a list of tax suppliers with all the relevant fields in the output flist.

This opcode is not called by any opcode.

Retrieving Tax Supplier Data

To retrieve tax supplier data, use PCM_OP_RATE_POL_MAP_TAX_SUPPLIER. You can customize this opcode to change how a tax supplier is derived for a specific BRM event.

This opcode is called by PCM_OP_BILL_TAX_EVENT.

This opcode can function in either of two ways:

- If a **tax_supplier_map** file is used with a corresponding entry in the CM **pin.conf** file, this opcode searches the **tax_supplier_map** file by company ID based on the charge offer name and ship to categories. If a matching entry is found, it returns that entry from the **tax_supplier_map** file. If no matching entry exists, the opcode returns **NULL** as the address of the flist.
- If the **tax_supplier_map** file does not exist or the CM **pin.conf** entry does not specify using a **tax_supplier_map** file, this opcode tries to find the **tax_supplier** object POID in the PIN_FLD_PRODUCTS array of the account object.

PCM_OP_RATE_POL_MAP_TAX_SUPPLIER also retrieves the utility flag, which is used for telecommunication (telco) taxation, from the PIN_FLD_REGULATED_FLAG field of the **/profile/tax_supplier** object and returns it on the output flist.

Retrieving Tax Location Data

To customize the way PCM_OP_RATE_EVENT obtains the locations for tax jurisdictions, use PCM_OP_RATE_POL_TAX_LOC. This opcode returns the locations for an event, which are then used to establish jurisdictions for tax calculation.

> **Note:**
>
> For a telephony event, the locations contain additional information, which is enclosed in square ([]) brackets.

For example, you can obtain the value for the PIN_FLD_ORDER_ACCEPT field from a different source. Additionally, you can provide a geocode instead of an NPA/NXX for telephony events.

PCM_OP_RATE_POL_TAX_LOC takes an account POID as input and returns these fields:

- PIN_FLD_SHIP_TO: The ship-to address; for telephony events, the call termination number.
- PIN_FLD_SHIP_FROM: The ship-from address; for telephony events, the call origination number.
- PIN_FLD_ORDER_ACCEPT: The order accept address; for telephony events, the charge-to number.
- PIN_FLD_ORDER_ORIGIN: The order origin address.

It returns them in four strings with this syntax:

```
city; state_abbreviation;zipcode; country;[code,location,international_indicator]
```

where:

- *code* is the geocode or NPA/NXX.
- *location* specifies the type of code:
  - **0**: Address
  - **1**: Geocode
  - **2**: NPA/NXX
- *international_indicator* specifies the type of call:
  - **1**: North America numbering plan
  - **2**: North America originated, overseas terminated
  - **3**: North America originated, overseas terminated and billed
  - **4**: IOC North America originated, overseas terminated
  - **5**: Overseas originated, North America terminated
  - **6**: IOC North America originated, overseas terminated and billed
  - **7**: Overseas originated, overseas terminated

For example:

```
1  PIN_FLD_SHIP_TO            STR [0] "Cupertino; CA; 95014; USA;[408572,2,1]"
1  PIN_FLD_SHIP_FROM          STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_ORDER_ORIGIN       STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_ORDER_ACCEPT       STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_LOCATION_MODE     ENUM [0] 1
1  PIN_FLD_INTERNATIONAL_IND  INT [0] 1
```

By default, PCM_OP_RATE_POL_TAX_LOC uses the sources for locations described in Table 23-2:

Table 23-2 PCM_OP_RATE_POL_TAX_LOC Source Locations

Location	Default Source
PIN_FLD_SHIP_TO	The account billing address, obtained from the PIN_FLD_NAMEINFO array of the **/account** object.
PIN_FLD_SHIP_FROM	Same as PIN_FLD_ORDER_ACCEPT.
PIN_FLD_ORDER_ACCEPT	The value for the **fm_rate_pol provider_loc** entry in the CM **pin.conf** file. For information on the default PIN_FLD_ORDER_ACCEPT location, see the **fm_rate_pol provider_loc** entry in the CM **pin.conf** file.
PIN_FLD_ORDER_ORIGIN	Same as PIN_FLD_ORDER_ACCEPT.

Retrieving Additional Tax Data from Vertex Communications Tax Q Series

When you perform tax calculation by using Vertex Communications Tax Q Series, it returns tax data in an input flist to the BRM API. The tax data is then processed and stored in the BRM database. For a list of data that Vertex Communications Tax Q Series returns to the BRM API by default, see "Default Tax Data Returned by Vertex Communications Tax Q Series" for more information.

You can also customize BRM to request and store additional data from Vertex Communications Tax Q Series by performing the following:

1. Requesting additional data from Vertex Communications Tax Q Series.

   See "Requesting Additional Data from Vertex Communications Tax Q Series" for more information.

2. Retrieving the additional tax data returned by Vertex Communications Tax Q Series.

   See "Requesting Additional Data from Vertex Communications Tax Q Series" for more information.

3. Storing the additional tax data in the BRM database.

   See "Storing the Requested Vertex Tax Data in the BRM Database" for more information.

Default Tax Data Returned by Vertex Communications Tax Q Series

Vertex Communications Tax Q Series makes its attributes available to other applications through data handles. BRM uses tax data from the following two data handles only:

- The Register Transaction data handle, which defines tax transaction attributes.
- The Register Transaction Tax Detail data handle, which defines tax jurisdiction attributes.

Vertex Communications Tax Q Series sends tax attributes to the BRM API in the following input flist structures:

- **PIN_FLD_TAXES input flist array**: Contains the result of a tax calculation at the transaction level.
- **PIN_FLD_TAXES.PIN_FLD_SUBTOTAL input flist array**: Contains the result of a tax calculation at the jurisdiction level, such as at the Federal level or at the State level. In each jurisdiction, the taxes are broken into subtypes, such as 911 and DEAF.

Table 23-3 shows the tax attributes that are sent to the BRM API by default and the flist structure in which the attribute is passed:

Table 23-3 Default BRM Tax Attributes and flist Structure

Vertex Communications Tax Q Series Data Handle	Vertex Communications Tax Q Series Attributes	BRM Input Flist Structure
Register Transaction data handle	• eCtqAttribChargeToLocationMode • eCtqAttribChargeToPostalCode • eCtqAttribTaxedGeoCodeOverrideCode • eCtqAttribInvoiceDate • eCtqAttribBilledLines • eCtqAttribInvoiceNumber • eCtqAttribServiceCode • eCtqAttribCategoryCode • eCtqAttribCreditCode • eCtqAttribDescriptionFlag • eCtqAttribTaxedGeoCodeIncorporatedCode • eCtqAttribFederalExemptFlag • eCtqAttribStateExemptFlag • eCtqAttribCountyExemptFlag • eCtqAttribCityExemptFlag • eCtqAttribWriteJournal **Note:** The eCtqAttribWriteJournal attribute is retrieved from the Register Subsystem data handle.	PIN_FLD_TAXES array
Register Transaction Tax Detail data handle	• eCtqAttribTrunksTaxed • eCtqAttribTaxCode • eCtqAttribLinkTaxAmount • eCtqAttribRowCount	PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array

Requesting Additional Data from Vertex Communications Tax Q Series

You can request additional tax data from Vertex Communications Tax Q Series by customizing PCM_OP_RATE_POL_PRE_TAX. Vertex Communications Tax Q Series returns the requested tax data in the input flist to PCM_OP_RATE_POL_POST_TAX.

To request additional tax data:

- Customize PCM_OP_RATE_POL_PRE_TAX to request data from the following two Vertex data handles: Register Transaction data handle and Register Transaction Tax Detail data handle.
- Add custom output flist fields to PCM_OP_RATE_POL_PRE_TAX. To request additional tax data, add the flist fields shown in bold below to the PIN_FLD_RESULTS output flist array:

```
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1   PIN_FLD_TAXES                 [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES     ARRAY [ArrayIndexValue] allocated 20, used 2
3       PIN_FLD_TYPE           ENUM [0] VertexDataType
1   PIN_FLD_SUBTOTAL              [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES     ARRAY [ArrayIndexValue] allocated 20, used 2
3       PIN_FLD_TYPE           ENUM [0] VertexDataType
```

  where:

  – The PIN_FLD_TAXES structure specifies that you are requesting data from the Vertex Register Transaction data handle. All elements under this array must reference attributes from the Register Transaction data handle only.

  - The PIN_FLD_SUBTOTAL structure specifies that you are requesting data from the Vertex Register Transaction Tax Detail data handle. All elements under this array must reference attributes from the Register Transaction Tax Detail data handle only.
  - *ArrayIndexValue* specifies the Vertex Communications Tax Q Series attribute ID that you are requesting. For example, set the PIN_FLD_FIELD_NAMES array index to **304** to request the eCtqAttribTrunksTaxed attribute.
  - *VertexDataType* specifies the Vertex data type ID for the Vertex Communications Tax Q Series attribute you are requesting. Supported Vertex data types include the following:
    * eCtqInt
    * eCtqString
    * eCtqBool
    * eCtqFloat
    * eCtqDouble
    * eCtqLong
    * eCtqDate

For example, the following shows custom output flist fields for requesting additional tax data:

```
0 PIN_FLD_RESULTS               ARRAY [0] allocated 20, used 1
1   PIN_FLD_TAXES                     [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES       ARRAY [271] allocated 20, used 2
3       PIN_FLD_TYPE             ENUM [0] 3
1   PIN_FLD_SUBTOTAL                  [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES       ARRAY [285] allocated 20, used 2
3       PIN_FLD_TYPE             ENUM [0] 10
```

Retrieving Additional Tax Data

The additional data you request from Vertex Communications Tax Q Series is returned in the PCM_OP_RATE_POL_POST_TAX input flist. You can customize the BRM API to retrieve the tax data from the input flist fields and process it according to your business needs.

Vertex Communications Tax Q Series returns the additional data in the following PCM_OP_RATE_POL_POST_TAX input flist structure under the PIN_FLD_TAXES array or the PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array:

- If passed in the PIN_FLD_TAXES array, the data is for one tax transaction. This information is from the Register Transaction data handle.
- If passed in the PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array, the tax data is for one jurisdiction. This information is from the Register Transaction Tax Detail data handle.

```
1  PIN_FLD_FIELD_NAMES           ARRAY [ArrayIndexValue] allocated 20, used 3
2    PIN_FLD_TYPE                      ENUM [0] VertexDataType
2    PIN_FLD_VERTEX_CTQ_BRMDataType    INT [0] BRMTaxData
```

where:

  - *ArrayIndexValue* specifies the Vertex Communications Tax Q Series attribute ID. For example, enter **295** to request data from the eCtqAttribTaxedGeoCode attribute.
  - *VertexDataType* specifies the data type ID of the Vertex Communications Tax Q Series attribute. For example, enter **3** for the eCtqInt data type.

- *BRMTaxData* is the tax data from the specified Vertex Communications Tax Q Series attribute.
- PIN_FLD_VERTEX_CTQ_*BRMDataType* is the name of the input flist field that contains the tax data. Table 23-4 specifies the name of the BRM input flist field that corresponds to each Vertex data type.

Table 23-4 BRM Input flist for Vertex Data Type

BRM Data Type	Vertex Data Type
INT	eCtqInt
STR	eCtqString
BOOL	eCtqBool
FLOAT	eCtqFloat
DOUBLE	eCtqDouble
LONG	eCtqLong
STR **Important:** This field will contain the date in Vertex Communications Tax Q Series format (*CCYYMMDD*). This is different from the mapping for custom input fields, in which **eCtqDate** is mapped to PIN_FLD_VERTEX_CTQ_TIMESTAMP.	eCtqDate

For example, the following shows a simple PCM_OP_RATE_POL_POST_TAX input flist. The flist fields for the additional tax data are shown in bold.

```
# number of field entries allocated 20, used 2
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /account 27225 0
0 PIN_FLD_EVENT_OBJ            POID [0] 0.0.0.1 /event/billing/cycle/tax 11111 0
0 PIN_FLD_TAXES               ARRAY [0] allocated 20, used 3
1   PIN_FLD_TAXPKG_TYPE         ENUM [0] 0
1   PIN_FLD_TAX              DECIMAL [0] 0.70
1   PIN_FLD_FIELD_NAMES        ARRAY [295] allocated 20, used 3  // Array index =
eCtqAttribTaxedGeoCode
2     PIN_FLD_TYPE              ENUM [0] 10
2     PIN_FLD_VERTEX_CTQ_STR     STR [0] "123456789"
1   PIN_FLD_SUBTOTAL           ARRAY [0] allocated 20, used 10
2     PIN_FLD_TAX              DECIMAL [0] 0.70
2     PIN_FLD_TYPE                ENUM [0] 0
2     PIN_FLD_NAME                 STR [0] "US; CA; Sunnyvale; ; 94086"
2     PIN_FLD_AMOUNT_GROSS     DECIMAL [0] 19.88
2     PIN_FLD_PERCENT          DECIMAL [0] 0.035000
2     PIN_FLD_AMOUNT_TAXED     DECIMAL [0] 19.88
2     PIN_FLD_AMOUNT_EXEMPT    DECIMAL [0] 0.00
2     PIN_FLD_SUBTYPE             ENUM [0] 0
2     PIN_FLD_DESCR                STR [0] "Excise"
2     PIN_FLD_LOCATION_MODE       ENUM [0] 0
2     PIN_FLD_FIELD_NAMES        ARRAY [304] allocated 20, used 3  // Array index =
eCtqAttribTrunksTaxed
3       PIN_FLD_TYPE              ENUM [0] 3
3       PIN_FLD_VERTEX_CTQ_INT     INT [0] 1
```

Each PIN_FLD_TAXES array element results in a single balance impact. All tax data from the PIN_FLD_TAXES array element is stored in the BRM **/event** object in the PIN_FLD_BAL_IMPACTS array.

Each PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array element results in a single balance impact for a jurisdiction. All tax data for that jurisdiction is stored in the BRM **/event** object in the PIN_FLD_TAX_JURISDICTIONS array.

> **Note:**
>
> Sometimes, a Vertex attribute may be missing from the PCM_OP_RATE_POL_POST_TAX input flist. This could be due to an exception condition. For example:
>
> - If taxes do not apply because of an undefined service code. In this case, the PIN_FLD_SUBTOTAL array will have a zero tax amount but none of the additional output fields.
> - If BRM requested a nonapplicable attribute. For example, if BRM requested the eCtqAttribOriginGeoCode attribute when calculating taxes for a wireless service based on the PPU location passed in as a ZIP code.

Storing the Requested Vertex Tax Data in the BRM Database

All additional Vertex Communications Tax Q Series data that you request is returned in the PCM_OP_RATE_POL_POST_TAX input flist. To customize the opcode to store the additional data in the BRM database:

- Create a custom tax storable class.

  See "Creating a Custom Storable Class for Vertex Tax Data" for more information.
- Customize PCM_OP_RATE_POL_POST_TAX to store the additional tax data in your custom tax storable class.

  See "Storing Vertex Tax Data in Your Custom Storable Class" for more information.
- Configure BRM to *itemize* taxes by jurisdiction.

Creating a Custom Storable Class for Vertex Tax Data

To store the additional data from Vertex Communications Tax Q Series, create a custom tax storable class, such as **/custom/tax/ctq_output**. The structure of the custom tax storable class should be similar to the following flist:

> **Note:**
>
> Make sure you define all fields in the custom tax storable class.

```
PIN_FLD_POID
PIN_FLD_ACCOUNT_OBJ
PIN_FLD_EVENT_OBJ
   PIN_FLD_CUSTOM_TAX_BAL_IMPACTS           ARRAY
      PIN_FLD_CUSTOM_TAX_TAXED_GEO_CODE     STR
...
   PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS     ARRAY
      PIN_FLD_ELEMENT_ID                    INT
      PIN_FLD_CUSTOM_TAX_TRUNKS_TAXED       INT
```

Storing Vertex Tax Data in Your Custom Storable Class

To store the additional Vertex Communications Tax Q Series tax data in your custom tax storable class, customize PCM_OP_RATE_POL_POST_TAX as follows:

1. Add the following compiler directives to include Vertex Communications Tax Q Series compiler definitions. Copy these files from the Vertex Communications Tax Q Series installation directory to the directory in which the policy source code resides.

   ```
   #include "stda.h"
   #include "ctqa.h"
   ```

2. In the policy source code directory, modify **Makefile** to add the **-DPORT_UNIXANSI Compiler** option.

   For example, you can use the CFLAGS variable as follows:

   ```
   CFLAGS += -DPORT_UNIXANSI
   ```

3. Create a new flist instance of your custom tax storable class and set PIN_FLD_EVENT_OBJ of this instance in the **/event** object.

4. In your custom tax storable class:

   - Add a PIN_FLD_CUSTOM_TAX_BAL_IMPACTS array element for each PIN_FLD_TAXES element passed in the PCM_OP_RATE_POL_POST_TAX input flist.
   - In the PIN_FLD_CUSTOM_TAX_BAL_IMPACTS element, set each custom tax storable class field to its corresponding value from the PIN_FLD_TAXES input flist array. For example, set the PIN_FLD_CUSTOM_TAX_TAXED_GEO_CODE storable class field to the value in the PIN_FLD_VERTEX_CTQ_STR input flist field.
   - Add a PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS array element for each PIN_FLD_TAXES.PIN_FLD_SUBTOTAL element passed in the PCM_OP_RATE_POL_POST_TAX input flist.
   - In the PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS element, set each custom tax storable class field to its corresponding value from the PIN_FLD_SUBTOTAL input flist array. For example, set the PIN_FLD_CUSTOM_TAX_TRUNKS_TAXED storable class field to the value in the PIN_FLD_VERTEX_CTQ_INT input flist field.
   - Link each PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS element to its corresponding parent PIN_FLD_CUSTOM_TAX_BAL_IMPACTS element by using the PIN_FLD_ELEMENT_ID field.

   > **Note:**
   >
   > BRM uses this same method to link PIN_FLD_TAX_JURISDICTIONS elements to corresponding parent PIN_FLD_BAL_IMPACTS elements in the **/event** object.

5. Create an instance of your custom tax storable class in the BRM database by calling PCM_OP_CREATE_OBJ.

6. Drop the PIN_FLD_FIELD_NAMES arrays from PCM_OP_RATE_POL_POST_TAX output flist, because they are no longer needed.

Note:

- In the rare case that no taxes are computed, the policy input flist will include a single PIN_FLD_TAXES element set to 0 and no PIN_FLD_SUBTOTAL elements. Likewise, when a jurisdiction check is performed to validate the account address, the policy input flist will not include a PIN_FLD_TAXES element. In both cases, you do not need to create a new instance of your custom tax storable class.
- For event-time taxation, the PCM_OP_RATE_POL_POST_TAX input flist does not include the PIN_FLD_EVENT_OBJ field.
- For reporting purposes, you can relate a taxed event's default tax fields with the additional tax fields by using the **/event** POID as the join criterion between the **/event** object and your custom tax object.

24

Using the IP Address Manager APIs

Learn about the tasks you can perform using the Oracle Communications Billing and Revenue Management (BRM) IP Address Manager APIs.

For an overview of IP Address Manager features, see "About IP Address Manager" in *BRM Telco Integration*.

Topics in this document:

- Opcodes Described in This Chapter
- Managing Your IP Address Device Life Cycle
- Managing your APN Device Life Cycle
- Extending the IP Address Manager Storable Classes
- Adding Business Logic to the IP Address and APN Policy FMs

Opcodes Described in This Chapter

Table 24-1 lists the opcodes described in this chapter.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Table 24-1 Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_APN_POL_DEVICE_ASSOCIATE	Associating APN with an Account or Service
PCM_OP_APN_POL_DEVICE_CREATE	Creating an APN Device Changing the APN Device State
PCM_OP_APN_POL_DEVICE_DELETE	Deleting an APN Device
PCM_OP_APN_POL_DEVICE_SET_ATTR	Modifying an APN Device
PCM_OP_APN_POL_DEVICE_SET_STATE	Changing the APN Device State
PCM_OP_DEVICE_ASSOCIATE	Associating an IP Address with Accounts or Services Associating APN with an Account or Service
PCM_OP_DEVICE_CREATE	Creating a Single IP Address Device Creating an APN Device
PCM_OP_DEVICE_DELETE	Deleting an IP Address Device Deleting an APN Device

Table 24-1 (Cont.) Opcodes Described in This Chapter

Opcode	Topic
PCM_OP_DEVICE_POL_ASSOCIATE	Disassociating an IP Address Device from Accounts or Services Associating APN with an Account or Service
PCM_OP_DEVICE_POL_CREATE	Creating a Single IP Address Device Creating an APN Device
PCM_OP_DEVICE_POL_DELETE	Deleting an APN Device
PCM_OP_DEVICE_POL_SET_ATTR	Modifying an IP Address Device Modifying an APN Device
PCM_OP_DEVICE_POL_SET_STATE	Changing IP Device States from Unallocated to Returned Changing the APN Device State
PCM_OP_DEVICE_SET_ATTR	Modifying an IP Address Device' Modifying an APN Device
PCM_OP_DEVICE_SET_STATE	Changing IP Device States from Unallocated to Returned Changing the APN Device State
PCM_OP_IP_DEVICE_CREATE	Creating a Single IP Address Device Creating a Range of IP Address Devices Creating a Range of IP Addresses with a Subnet Mask
PCM_OP_IP_DEVICE_DELETE	Deleting an IP Address Device
PCM_OP_IP_DEVICE_SET_ATTR	Modifying an IP Address Device
PCM_OP_IP_DEVICE_SET_STATE	Changing IP Device States from Unallocated to Returned
PCM_OP_IP_POL_DEVICE_ASSOCIATE	Associating an IP Address with Accounts or Services
PCM_OP_IP_POL_DEVICE_CREATE	Creating a Single IP Address Device Customizing IP Address Creation Changing the APN Device State
PCM_OP_IP_POL_DEVICE_DELETE	Deleting an IP Address Device
PCM_OP_IP_POL_DEVICE_SET_ATTR	Modifying an IP Address Device
PCM_OP_IP_POL_DEVICE_SET_STATE	Changing IP Device States from Unallocated to Returned Modifying an IP Address Device

Managing Your IP Address Device Life Cycle

This section explains how to use the IP Facilities Module (FM) and IP Policy FM to manage your IP address inventory.

Creating a Single IP Address Device

You create a single IP address device by calling PCM_OP_IP_DEVICE_CREATE and passing it an flist that includes:

- A type-only POID that specifies the device type of **/device/ip** in the PIN_FLD_POID field
- A valid IP address in the PIN_FLD_START_ADDRESS field
- The database being used

You add any validation checks or other business logic related to creating a single IP address device to PCM_OP_IP_POL_DEVICE_CREATE.

This is the calling sequence:

1. PCM_OP_IP_DEVICE_CREATE performs these tasks:
   - Verifies that the device type is **/device/ip**
   - Returns an error if the IP address already exists
   - Calls PCM_OP_DEVICE_CREATE
2. PCM_OP_DEVICE_CREATE performs this task:
   - Calls PCM_OP_DEVICE_POL_CREATE
3. PCM_OP_DEVICE_POL_CREATE performs this task:
   - Passes everything to PCM_OP_IP_POL_DEVICE_CREATE
4. PCM_OP_IP_POL_DEVICE_CREATE performs these tasks:
   - Determines the IP address number
   - Validates that the new IP/APN device combination is unique in the database
   - Runs any other validation checks or business logic that you have added
   - Returns an error if the IP device/APN device combination is a duplicate
   - Passes this information back to PCM_OP_DEVICE_CREATE
5. PCM_OP_DEVICE_CREATE performs these tasks:
   - Creates the object
   - Returns an error if the address is invalid

For more information on using PCM_OP_DEVICE_CREATE and PCM_OP_DEVICE_POL_CREATE, see "Creating /device Objects".

Creating a Range of IP Address Devices

You create a range of IP addresses the same way you create a single IP address, with a single call to PCM_OP_IP_DEVICE_CREATE. To create a range of addresses you enter valid start and end IP addresses in the PIN_FLD_START_ADDRESS and PIN_FLD_END_ADDRESS fields. This opcode creates a contiguous range of **/device/ip** objects, one for each of the addresses, including the start and end addresses.

You have the option of using wildcard characters in the last two octets of the PIN_FLD_START_ADDRESS field. If they are used, PCM_OP_IP_DEVICE_CREATE creates a range using them.

If wildcard characters are not used, PCM_OP_IP_DEVICE_CREATE performs one of the following actions:

- Creates a range using the end address
- Creates a range using subnet mask
- Creates a single IP device

PCM_OP_IP_DEVICE_CREATE returns an error if the range is invalid.

Creating a Range of IP Addresses with a Subnet Mask

You create a range of all possible IP addresses under a specific subnet mask the same way you create a single IP address, with a single call to PCM_OP_IP_DEVICE_CREATE. To create an IP address/subnet mask range, you pass in the following information:

- A valid address in the PIN_FLD_START_ADDRESS field
- A valid subnet mask address in the PIN_FLD_SUBNET_MASK field

PCM_OP_IP_DEVICE_CREATE verifies that the range is valid and then creates all possible IP addresses in the range including the start and end addresses.

PCM_OP_IP_DEVICE_CREATE returns an error if the range is invalid.

Customizing IP Address Creation

Use PCM_OP_IP_POL_DEVICE_CREATE to customize IP address device creation. By default this opcode only checks to make sure the IP addresses to be created are not duplicates of existing IP addresses. This opcode is an empty hook.

Associating an IP Address with Accounts or Services

You associate an IP address with any number of accounts or services by calling PCM_OP_DEVICE_ASSOCIATE. You also make device state changes to or from the **allocated** state with a call to PCM_OP_DEVICE_ASSOCIATE.

PCM_OP_DEVICE_ASSOCIATE requires the this information:

- An array of IP address objects to associate
- A PIN_FLD_SERVICES array containing an **/account** or **/service** POIDs in the PIN_FLD_ACCOUNT_OBJ or PIN_FLD_SERVICE_OBJ field
- The PIN_FLD_FLAGS field set to **0** (indicating an association)

Use PCM_OP_IP_POL_DEVICE_ASSOCIATE to customize how to verfiy changing the state of an IP address device when services are assigned to, or removed from the device.

Disassociating an IP Address Device from Accounts or Services

The process of disassociating an IP address from accounts or services is much like associating IP addresses and accounts or services, except that you send in the input flist with PIN_FLD_FLAGS set to **1**. In this case PCM_OP_DEVICE_POL_ASSOCIATE:

- Confirms that none of the IP devices are associated with a service or account
- Changes the states from **allocated** to **unallocated**
- Disassociates the accounts or services by removing them from the **/device/ip** object

Changing IP Device States from Unallocated to Returned

You change IP device states between **unallocated** to **returned** using a call to PCM_OP_IP_DEVICE_SET_STATE. You pass in an array of the IP devices that get the state change, and the new state.

> **Note:**
>
> To make any device state changes to or from the **allocated** state, see "Associating an IP Address with Accounts or Services".

PCM_OP_IP_DEVICE_SET_STATE calls PCM_OP_DEVICE_SET_STATE.

When setting states for multiple IP devices, PCM_OP_IP_DEVICE_SET_STATE operates on a contiguous range of IP addresses devices based on starting and ending IP addresses you supply. The opcode sets the state for all **/device/ip** objects in that range.

You add any validation checks or other business logic related to changing the state of all instances of an IP device to PCM_OP_IP_POL_DEVICE_SET_STATE.

PCM_OP_IP_POL_DEVICE_SET_STATE enables you to customize how to verify that an IP address device is eligible to accept a device state change.

PCM_OP_IP_POL_DEVICE_SET_STATE changes the state for all instances of an IP device. This opcode searches for all instances of an IP address in the database and changes their states. If the IP device is associated with multiple APNs, they are all changed.

PCM_OP_IP_POL_DEVICE_SET_STATE is called by PCM_OP_DEVICE_POL_SET_STATE.

PCM_OP_IP_POL_DEVICE_SET_STATE determines whether the IP state change is from or to the **allocated** state.

- If the state change is from or to allocated, it calls PCM_OP_IP_POL_DEVICE_ASSOCIATE to perform the change.
- For all other state changes, it calls PCM_OP_IP_DEVICE_SET_STATE to make the state change.

Modifying an IP Address Device

You use PCM_OP_IP_DEVICE_SET_ATTR to change the APN an IP is associated with. You call this opcode and pass it the POID of the **/device/ip** object and the change you want to make.

This opcode calls PCM_OP_DEVICE_SET_ATTR. To set attributes for a range of devices, send in start and end POIDs and this opcode make changes to all IP devices within that contiguous range.

You add any validation checks or other business logic related to modifying an IP device to PCM_OP_IP_POL_DEVICE_SET_ATTR.

PCM_OP_IP_POL_DEVICE_SET_ATTR verifies that an IP address device is not in an **allocated** or **returned** state, and then changes an attribute for the **/device/ip** object.

PCM_OP_IP_POL_DEVICE_SET_ATTR is an empty hook.

PCM_OP_IP_POL_DEVICE_SET_ATTR is called by PCM_OP_DEVICE_POL_SET_ATTR and PCM_OP_IP_POL_DEVICE_SET_STATE.

Sorting IP Devices by Using Canonical IP Address

The **/device/ip** object stores a canonical version of the device IP address that sorts more logically than a raw IP address. The PIN_FLD_DEVICE_CANONICAL_ID field contains an IP

address that has been canonicalized by expanding all four parts to three digits each by using **0** (zero) as a placeholder. For example, the IP address 152.3.44.67 is stored in PIN_FLD_DEVICE_CANONICAL_ID as 152.003.044.067. Any standard ASCII search will sort these normalized IP address representations in simple numerical order.

Deleting an IP Address Device

To delete an IP address, call PCM_OP_IP_DEVICE_DELETE with a **/device/ip** object on the input flist. Before you delete IP devices, be sure to disassociate them from and accounts, services, or APNs.

This opcode calls PCM_OP_IP_DEVICE_DELETE. This opcode deletes one IP device at a time. You must call it once for each device to delete.

You add any validation checks or other business logic related to deleting an IP device to PCM_OP_IP_POL_DEVICE_DELETE.

PCM_OP_IP_POL_DEVICE_DELETE verifies that an IP address device is not in an **allocated** state, and then deletes the **/device/ip** object.

PCM_OP_IP_POL_DEVICE_DELETE is an empty hook.

You cannot use the IP Address Administrator to delete devices. Using PCM_OP_DEVICE_DELETE is the only way to do this.

Managing your APN Device Life Cycle

This section explains how to use the APN FM to manage your APN device inventory.

Creating an APN Device

To create an APN device, call PCM_OP_DEVICE_CREATE with a **/device/apn** object on the input flist. You add any validation checks or other business logic related to creating an APN device to PCM_OP_APN_POL_DEVICE_CREATE.

PCM_OP_APN_POL_DEVICE_CREATE is called by PCM_OP_DEVICE_POL_CREATE.

PCM_OP_APN_POL_DEVICE_CREATE verifies that the APN names and IDs being created are correct and determines whether there are any existing APN devices with duplicate names.

PCM_OP_APN_POL_DEVICE_CREATE is an empty hook.

Associating APN with an Account or Service

You associate an APN with an account or service by calling PCM_OP_DEVICE_ASSOCIATE, passing it an flist with the following:

- The POID of the **/device/apn** object in the PIN_FLD_POID field
- An account or service POID in the PIN_FLD_ACCOUNT_OBJ or PIN_FLD_SERVICE_OBJ field

You add any validation checks or other business logic related to associating an APN with an account or service PCM_OP_APN_POL_DEVICE_ASSOCIATE.

PCM_OP_APN_POL_DEVICE_ASSOCIATE is called by PCM_OP_DEVICE_POL_ASSOCIATE.

PCM_OP_APN_POL_DEVICE_ASSOCIATE verifies that the APN device is in a **new** or **usable** state.

For more information on associating devices by using PCM_OP_DEVICE_ASSOCIATE and PCM_OP_DEVICE_POL_ASSOCIATE, see "Device Management Opcodes", "Device FM Policy Opcodes", and "Device FM Standard Opcodes".

Modifying an APN Device

To modify an APN, call PCM_OP_DEVICE_SET_ATTR, passing it a valid APN ID in the PIN_FLD_DEVICE_ID field. You add any validation checks or other business logic related to modifying an APN device to PCM_OP_APN_POL_DEVICE_SET_ATTR.

PCM_OP_APN_POL_DEVICE_SET_ATTR verifies that an APN device is in a state to accept changes, and then makes the changes.

PCM_OP_APN_POL_DEVICE_SET_ATTR confirms the following:

- The APN device object type is **/device/apn**.
- None of the IP addresses associated with the APN device are in an **allocated** state.

PCM_OP_APN_POL_DEVICE_SET_ATTR is called by PCM_OP_DEVICE_POL_SET_ATTR.

For more information on modifying devices using PCM_OP_DEVICE_SET_ATTR and PCM_OP_DEVICE_POL_SET_ATTR, see "Device Management Opcodes", "Device FM Policy Opcodes", and "Device FM Standard Opcodes".

Changing the APN Device State

To change the state of an APN device, pass the **/device/apn** object POID and the new state to PCM_OP_DEVICE_SET_STATE. Add any validation checks or other business logic related to changing the state of an APN to PCM_OP_APN_POL_DEVICE_SET_STATE.

PCM_OP_APN_POL_DEVICE_SET_STATE verifies that an APN device is eligible to accept a device state change, and then makes the changes.

PCM_OP_APN_POL_DEVICE_SET_STATE is called by PCM_OP_DEVICE_POL_SET_STATE.

By default, PCM_OP_APN_POL_DEVICE_SET_STATE confirms the following:

- If the state change is from **usable** to **unusable**, none of its associated IP devices are in an **allocated** state. If any are, an error is returned, and the entire transaction is rolled back.
- If the APN device state change is from **new** to **usable**, PCM_OP_IP_POL_DEVICE_CREATE was used to call this opcode.

When changing device states:

- The **new** device state is set by PCM_OP_APN_POL_DEVICE_CREATE only when an APN is created.
- Changing the device state from **new** to **usable** is allowed only as part of creating an IP device (which must be associated with an APN. In this case PCM_OP_APN_POL_DEVICE_SET_STATE confirms that PCM_OP_IP_POL_DEVICE_CREATE is part of the calling sequence. See "Creating a Single IP Address Device" and "Creating a Range of IP Address Devices".

Deleting an APN Device

You delete an APN device by calling PCM_OP_DEVICE_DELETE with the object type of **/device/apn** in the input flist. Before you delete the APN device, be sure to disassociate any IP devices or services. You add any validation checks or other business logic related to deleting an APN device to PCM_OP_APN_POL_DEVICE_DELETE.

PCM_OP_APN_POL_DEVICE_DELETE performs validation checks and deletes associated IP devices during APN device deletion. If any associated IP devices are allocated, this opcode does not delete it or the APN device.

PCM_OP_APN_POL_DEVICE_DELETE is called by PCM_OP_DEVICE_POL_DELETE.

You cannot use the IP Address Administrator application to delete devices. Using PCM_OP_DEVICE_DELETE is the only way to do this.

Extending the IP Address Manager Storable Classes

You use the BRM Storable Class Editor to extend the **/device/ip** storable class to meet your business needs. For more information, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

Adding Business Logic to the IP Address and APN Policy FMs

The IP and APN policy Facilities Modules (FMs) are designed for you to add your own customizations, such as additional validation checks. For information on extending these FMs, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.