Oracle® Communications Billing and Revenue Management

Developer's Reference

Release 7.5

E16714-17

April 2017

ORACLE®

Oracle Communications Billing and Revenue Management Developer's Reference, Release 7.5

E16714-17

Contents

2 PIN Libraries Reference

8 Sample Applications

Preface

This book provides reference information for Oracle Communications Billing and Revenue Management (BRM) application programming interfaces (APIs).

Audience

This document is intended for developers.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Help Center:

http://docs.oracle.com

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

https://edelivery.oracle.com

Documentation Accessibility

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

Access to Oracle Support

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

Document Revision History

The following table lists the revision history for this book.

Version	Date	Description
E16714-01	November 2011	Initial release.

Version	Date	Description
E16714-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ■ Made minor formatting and text changes. ■ Documented the following new policy opcodes: PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE ■ Documented the following new standard opcodes: PCM_OP_CUST_GET_LIFECYCLE_STATES PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE
E16714-03	August 2012	Documentation updates for BRM 7.5 Patch Set 2. ■ Documented the following new policy opcodes: PCM_OP_COLLECTIONS_POL_ASSIGN_DCA PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY ■ Documented the following new standard opcodes: PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER PCM_OP_COLLECTIONS_GROUP_CREATE PCM_OP_COLLECTIONS_GROUP_DELETE PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO PCM_OP_COLLECTIONS_GROUP_MODIFY PCM_OP_COLLECTIONS_GROUP_SET_PARENT PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY PCM_OP_COLLECTIONS_REPLACE_SCENARIO PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS

Version	Date	Description
E16714-04	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ■ Documented the following new policy opcodes: PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE ■ Documented the following new standard opcodes: PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE ■ Documented the following new notification events in "Event Notification Definitions": **/event/notification/offer_profile/create** **/event/notification/offer_profile/update** **/event/notification/offer_profile/delete** **/event/notification/offer_profile/ThresholdBreach**
E16714-05	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ■ Replaced multidatabase information with multischema information.
E16714-06	July 2013	Documentation updates for BRM 7.5 Patch Set 5. ■ Documented the following new standard opcode: PCM_OP_AR_BILL_CREDIT_TRANSFER
E16714-07	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16714-08	October 2013	Documentation updates for BRM 7.5 Patch Set 6. ■ Made minor formatting and text changes. ■ Documented the following standard opcode: PCM_OP_TCF_AAA_REFUND
E16714-09	May 2014	Documentation updates for BRM 7.5 Patch Set 8. ■ Updated the transaction handling information for PCM_OP_CUST_COMMIT_CUSTOMER. ■ Made minor updates to the following opcodes to support XA transactions: PCM_OP_TRANS_ABORT PCM_OP_TRANS_COMMIT PCM_OP_TRANS_OPEN

Version	Date	Description
E16714-10	August 2014	Documentation updates for BRM 7.5 Patch Set 9. ■ Made minor formatting and text changes in Chapter 1, "Opcode Reference". ■ Made minor updates to the following opcodes for trial invoicing: PCM_OP_BILL_MAKE_BILL PCM_OP_BILL_MAKE_TRIAL_BILL PCM_OP_INV_POL_PREP_INVOICE ■ Documentation added for RADIUS Manager. ■ Added the following new opcodes for SEPA payment processing: PCM_OP_AR_REVERSE_REFUND PCM_OP_CUST_AMEND_CREDITOR_INFO PCM_OP_CUST_AMEND_MANDATE PCM_OP_CUST_CANCEL_MANDATE
E16714-11	October 2014	Documentation updates for BRM 7.5 Patch Set 10. ■ Modified the description of the "PCM_OP_AR_REVERSE_WRITEOFF" opcode for write-off reversals at the bill-unit level.
E16714-12	June 2015	Documentation updates for BRM 7.5 Patch Set 12. ■ Added the "PCM_OP_ACT_LOG_USER_ACTIVITY" opcode.
E16714-13	August 2015	Documentation updates for BRM 7.5 Maintenance Patch Set 1. ■ Modified the description of the "PCM_OP_CUST_POL_MODIFY_SERVICE" policy opcode. ■ Added the following new opcodes for News Feed: PCM_OP_ACT_POL_PROCESS_EVENTS PCM_OP_CUST_GET_NEWSFEED
E16714-14	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ■ Made minor formatting and text changes.
E16714-15	April 2016	Documentation updates for BRM 7.5 Patch Set 15. ■ Documented the following new standard opcodes: PCM_OP_BAL_GET_ECE_BALANCES PCM_OP_JOB_PROCESS_TEMPLATE
E16714-16	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ■ Modified the description of the "PCM_OP_RATE_POL_POST_RATING" policy opcode.
E16714-17	April 2017	Documentation updates for BRM 7.5 Patch Set 18. ■ Added the "pcmOpCatch" section. ■ Updated the "Opcode Calling Functions" section.

1

Opcode Reference

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) 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

For more information about opcode input and output flist specifications, see *BRM Opcode Flist Reference*.

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.

Account Synchronization FM Opcodes

The opcodes in Table 1–1 synchronize customer and service data with pipeline rating data.

Header File

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

Opcode Index

Table 1–1 Account Synchronization FM Opcodes

Opcode	Description	Use
PCM_OP_IFW_SYNC_PUBLISH_EVENT	Passes events associated with this opcode in your system's event notification list to the policy opcode. See the discussion on processing BRM events that make up account synchronization business events in *BRM Installation Guide*.	Limited
PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT	Policy for modifying the events passed to the standard opcode. See the discussion on modifying BRM events that make up account synchronization business events in *BRM Installation Guide*.	Recommended

PCM_OP_IFW_SYNC_PUBLISH_EVENT

Passes events associated with this opcode in your system's event notification list to the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode for processing.

By default, PCM_OP_IFW_SYNC_PUBLISH_EVENT passes events without modifying them.

See the following discussions in *BRM Installation Guide*:

- Configuring event notification for account synchronization
- Processing BRM events that make up account synchronization business events

PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT

Modifies events included in account synchronization business events.

Events that trigger event notification for account synchronization make up the business events that the Account Synchronization Data Manager (DM) sends to Pipeline Manager. This opcode modifies specified triggering events before they are published to Pipeline Manager.

This opcode can also be used to filter out certain events that are not included in account synchronization (for example, an event that has only balance impacts for currency resources), thereby reducing the traffic in the listener queue.

If you pass an event that does not need any modification to this opcode, the opcode passes that event to the EAI framework for publishing.

See the discussion on modifying BRM events that make up account synchronization business events in *BRM Installation Guide*.

Activity FM Policy Opcodes

The opcodes in Table 1–2 manage event creation, event recording, and event notification. Only the activity opcodes call the rating opcodes.

Header File

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

Opcode Index

Table 1–2 Activity FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE	Selects the bill to charge for events that occur between the end of a billing cycle and when billing applications are run. See the discussion on customizing how to bill events that occur between billing cycles. in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_ACT_POL_EVENT_LIMIT	Inactivates an account or account hierarchy, and sends a notification that a limit has been reached. See the discussion on inactivating accounts that exceed a specified limit in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_POL_EVENT_NOTIFY	By default, processes events for LDAP integration and email notification when invoked by event notification. Can be used to implement custom event notification operations. See the discussion on using event notification in *BRM Developer's Guide*.	Recommended
PCM_OP_ACT_POL_LOCK_SERVICE	Locks the service account after a specified number of invalid login attempts. See the discussion on changing the password in *BRM System Administrator's Guide*.	Recommended
PCM_OP_ACT_POL_LOG_USER_ACTIVITY	Adds additional details for the events that need to be logged. See the discussion on logging customer service representative activities in *BRM System Administrator's Guide*.	Recommended
PCM_OP_ACT_POL_POST_AUTHORIZE	Enables you to customize the final output flist for PCM_OP_ACT_AUTHORIZE. See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.	Recommended

Table 1–2 (Cont.) Activity FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ACT_POL_POST_REAUTHORIZE	Enables you to customize the final output flist for PCM_OP_ACT_REAUTHORIZE. See the discussion on how BRM reauthorizes prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_POL_PRE_AUTHORIZE	Enables you to customize the final input flist for PCM_OP_ACT_AUTHORIZE. See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_POL_PRE_REAUTHORIZE	Enables you to customize the final input flist for PCM_OP_ACT_REAUTHORIZE. See the discussion on how BRM reauthorizes prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_POL_PROCESS_EVENTS	Processes the events for which News Feed is enabled and updates the database with the corresponding News Feed entries. See the discussion on News Feed in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES	Scales multi-RUM quantities during prepaid authorization. See the discussion on scaling quantities for prepaid authorization requests in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_POL_SET_RESOURCE_STATUS	Sets the resource availability status, the scaled delay time, and the balance amount to be reserved during a quick authorization. See the discussion on overriding the traffic-light status, reservation amount, and scaled delay time in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS	Produces rating information for an event. See the discussion on customizing how to calculate RUMs in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_POL_SPEC_EVENT_CACHE	Defines what fields are to be cached. See the discussion on specifying event fields to cache for invoicing in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_ACT_POL_SPEC_GLID	Assigns a G/L ID to a partially rated or prerated event. See the discussion on assigning G/L IDs to pre-rated events in *BRM Configuring and Running Billing*.	Recommended

Table 1–2 (Cont.) Activity FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ACT_POL_SPEC_RATES	Defines a rate name for administrative events to be rated. See the discussion on assigning rate plans and impact categories to events in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_POL_SPEC_VERIFY	Specifies authorization checks for an action. See the discussion on customizing authentication checks in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_POL_VALIDATE_SCHEDULE	Provides a way for adding a policy check before creating a **/schedule** object for a deferred action. See the discussion on performing policy checks before scheduling deferred actions in *BRM Managing Customers*.	Recommended

PCM_OP_ACT_POL_CONFIG_BILLING_CYCLE

Selects the bill to charge for events that occur between the end of a billing cycle and when billing applications are run.

By default, this policy opcode selects the current month's bill, but you can customize this opcode to select the previous month's bill.

This opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on how to customize the billing events that occur between billing cycles in *BRM Configuring and Running Billing*.

PCM_OP_ACT_POL_EVENT_LIMIT

Inactivates an account or account hierarchy, and sends a notification that a limit has been reached.

This opcode is not called by any opcode.

See the discussion on inactivating accounts that exceed a specified limit in *BRM Managing Customers*.

PCM_OP_ACT_POL_EVENT_NOTIFY

Used by event notification operations.

By default, processes events for LDAP integration and email notification when invoked by event notification.

This opcode is called by various event notification processes and by the PCM_OP_ACT_POL_EVENT_LIMIT policy opcode.

See the discussion on using event notification and triggering custom operations in *BRM Developer's Guide*.

PCM_OP_ACT_POL_LOCK_SERVICE

Locks the service account after a specified number of invalid login attempts.

By default, this opcode locks the service account by calling PCM_OP_WRITE_FLDS. This is applicable only for **/service/pcm_client** and **/service/admin_client**.

See the discussion on changing the password in *BRM System Administrator's Guide*.

PCM_OP_ACT_POL_LOG_USER_ACTIVITY

Adds additional details for the events that need to be logged. To customize the **/user_activity** storable class, you can find additional event information under PIN_FLD_INHERITED_INFO in the input flist.

This opcode is called by the PCM_OP_ACT_LOG_USER_ACTIVITY opcode.

See the discussion on logging customer service representative activities in *BRM System Administrator's Guide*.

PCM_OP_ACT_POL_POST_AUTHORIZE

Enables you to customize the final output flist for PCM_OP_ACT_AUTHORIZE.

By default, does the following:

1. Receives an input flist from PCM_OP_ACT_AUTHORIZE.
2. Drops the PIN_FLD_RESULTS field added to the flist by PCM_OP_ACT_USAGE.
3. Copies the input flist to its output flist.
4. Returns its output flist to PCM_OP_ACT_AUTHORIZE.

This opcode is called by the PCM_OP_ACT_AUTHORIZE standard opcode.

See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.

PCM_OP_ACT_POL_POST_REAUTHORIZE

Enables you to customize the final output flist for PCM_OP_ACT_REAUTHORIZE.

By default, does the following:

1. Receives an input flist from PCM_OP_ACT_REAUTHORIZE.
2. Drops the PIN_FLD_RESULTS field added to the flist by PCM_OP_ACT_USAGE.
3. Copies the input flist to its output flist.
4. Returns its output flist to PCM_OP_ACT_REAUTHORIZE.

This opcode is called by the PCM_OP_ACT_REAUTHORIZE standard opcode.

See the discussion on how BRM reauthorizes prepaid services in *BRM Telco Integration*.

PCM_OP_ACT_POL_PRE_AUTHORIZE

Allows customization of the prepaid authorization process.

This opcode is called by the PCM_OP_ACT_AUTHORIZE standard opcode.

See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.

PCM_OP_ACT_POL_PRE_REAUTHORIZE

Allows customization of the prepaid reauthorization process.

This opcode is called by the PCM_OP_ACT_REAUTHORIZE standard opcode.

See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.

PCM_OP_ACT_POL_PROCESS_EVENTS

Processes the events for which News Feed is enabled and updates the database with the corresponding News Feed entries.

This opcode is called by the PCM_OP_ACT_PROCESS_EVENTS standard opcode.

See the discussion on News Feed in *BRM Managing Customers*.

PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES

Scales multi-RUM quantities during prepaid authorization.

This opcode is called by the PCM_OP_TCF_AAA_AUTHORIZE and PCM_OP_TCF_AAA_REAUTHORIZE standard opcodes.

See the discussion on scaling quantities for prepaid authorization requests in *BRM Telco Integration*.

PCM_OP_ACT_POL_SET_RESOURCE_STATUS

Sets the resource availability status, the scaled delay time, and the balance amount to be reserved during a quick authorization.

This opcode is called from the PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD opcode.

See the discussions on overriding the traffic-light status, reservation amount, and scaled delay time in *BRM Telco Integration*.

PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS

Defines the customized policy to specify the ratable usage metric (RUM) candidate.

This opcode is called by the PCM_OP_ACT_GET_CANDIDATE_RUMS standard opcode.

See the discussion on customizing how to calculate RUMs in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_POL_SPEC_EVENT_CACHE

Defines which balance impact fields are cached for invoicing.

Important: You can improve performance by limiting the amount of information cached. However, if you need the information, it is quicker to cache a field than have it read from the event table.

This opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on specifying event fields to cache for invoicing in *BRM Configuring and Running Billing*.

PCM_OP_ACT_POL_SPEC_GLID

Assigns a G/L ID to a prerated or partially rated event.

Based on the event type, this policy opcode retrieves a G/L ID from the **/config/map_glid** object.

All customization is done using **pin_glid** and CoA files.

This opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on assigning G/L IDs to pre-rated events in *BRM Configuring and Running Billing*.

PCM_OP_ACT_POL_SPEC_RATES

Specifies the rate plan and impact category for custom event attributes.

The opcode that calls PCM_OP_ACT_POL_SPEC_RATES passes in the inherited information for an event.

This opcode is not called by any opcode.

See the discussion on assigning rate plans and impact categories to events in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_POL_SPEC_VERIFY

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

This opcode is called by the PCM_OP_ACT_FIND_VERIFY standard opcode.

See the discussion on customizing authentication checks in *BRM Managing Customers*.

PCM_OP_ACT_POL_VALIDATE_SCHEDULE

Performs policy checks before creating **/schedule** objects. For example, you might write and include code to verify that an account balance is zero before scheduling it for closure.

By default, this opcode is an empty hook.

This opcode is called by the PCM_OP_ACT_SCHEDULE_CREATE standard opcode.

See the discussion on performing policy checks before scheduling deferred actions in *BRM Managing Customers*.

Activity FM Standard Opcodes

The opcodes in Table 1–3 manage event creation, event recording, and event notification. Only the activity opcodes call the rating opcodes.

Header File

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

Opcode Index

Table 1–3 Activity FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ACT_ACTIVITY	Logs an activity event and assesses any charges. See the discussion on rating and recording activity events in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_AUTHORIZE	Authorizes prepaid services. See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_CALC_MAX_USAGE	Calculates maximum available usage. See the discussion on about calculating the maximum available usage in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_CANCEL_AUTHORIZE	Cancels prepaid service authorizations and deletes their session and resource reservation information. See the discussion on how BRM cancels prepaid service authorization in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD	Reads prepaid service authorization and reauthorization threshold information from the **/config/auth_reauth_info** object. See the discussion on how BRM reduces authorization latencies and How BRM uses a scaled delay time to reduce network spikes during a tariff change in *BRM Telco Integration*.	Recommended

Table 1–3 (Cont.) Activity FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ACT_END_SESSION	For *prepaid* sessions, closes active sessions, releases their resources, gathers their information to store in the BRM database, deletes them from IMDB Cache, and ensures that they are rated and that their account balances are updated. For *postpaid* sessions, logs the end of a session event and assesses any charges. See the discussion on rating and recording session events in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_FIND	Finds a customer's account. See the discussion on finding a customer's account information in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_FIND_VERIFY	Authorizes a user to perform a specific action. See the discussion on authenticating customers by using your custom application in *BRM Managing Customers* and configuring the maximum number of invalid login attempts in *BRM System Administrator's Guide*.	Recommended
PCM_OP_ACT_GET_CANDIDATE_RUMS	Produces rating information for an event. See the discussion on generating ratable usage metrics (RUMs) in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_HANDLE_OOD_EVENT	Creates a rerate job. See the discussion on how BRM rerates out-of-order events in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_ACT_LOAD_SESSION	Creates, rates, and records a session event. See the discussion on creating opcodes for loading event data in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_LOG_USER_ACTIVITY	Logs user activities in the **/user_activity** storable class. See the discussion about logging CSR activities in *BRM System Administrator's Guide*.	Recommended
PCM_OP_ACT_LOGIN	Authorizes a user to start a session. See the discussion on starting sessions in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_LOGOUT	Records the end of a login session. See the discussion on recording the end of a session in *BRM Setting Up Pricing and Rating*.	Recommended

Table 1–3 (Cont.) Activity FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ACT_MULTI_AUTHORIZE	Authorizes or rates multiple prepaid services with a single call. See the discussion on authorizing multiple services for a user with a single call in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_REAUTHORIZE	Reauthorizes prepaid services. See the discussion on how BRM reauthorizes prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_ACT_SCHEDULE_CREATE	Creates a **/schedule** object. See the discussion on scheduling deferred actions in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_SCHEDULE_DELETE	Deletes a **/schedule** object. See the discussion on deleting deferred actions in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_SCHEDULE_EXECUTE	Executes deferred actions. See the discussion on executing deferred actions in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_SCHEDULE_MODIFY	Modifies a **/schedule** object. See the discussion on modifying deferred action descriptions in *BRM Managing Customers*.	Recommended
PCM_OP_ACT_START_SESSION	Creates prepaid and postpaid session events and records their start times. See the discussion on recording the start of a session in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ACT_UPDATE_SESSION	Updates session event information during prepaid and postpaid sessions. See the following discussions: ■ (Prepaid) How BRM updates prepaid sessions in *BRM Telco Integration* ■ (Prepaid) How BRM updates and reauthorizes prepaid sessions in *BRM Telco Integration* ■ (Postpaid) Updating a session event in *BRM Setting Up Pricing and Rating*	Recommended
PCM_OP_ACT_USAGE	Rates and records an event. See the discussion on how BRM rates and records usage events in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_ACT_ACTIVITY

Records and rates activity events.

See the discussion on rating and recording activity events in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_AUTHORIZE

Authorizes prepaid services.

If unsuccessful, this opcode returns a numerical code in the PIN_FLD_REASON field of the output flist. For a list of the reasons that correspond to these numeric codes, see the **pin_reserve.h** file.

If PIN_FLD_PIGGYBACK_FLAG is enabled in the input flist, this opcode returns the credit thresholds breached in the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist.

See the following discussions:

- How BRM authorizes users to access prepaid services in *BRM Telco Integration*
- Providing credit threshold breach information in in-session notifications for network connectivity applications in *BRM Telco Integration*

PCM_OP_ACT_CALC_MAX_USAGE

Calculates the maximum available usage based on one of the following:

- The account's credit limit and current balance.
- The amount reserved for the session.

See the discussion on about calculating the maximum available usage in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_CANCEL_AUTHORIZE

Cancels prepaid service authorizations. Depending on the value in the input PIN_FLD_DELETED_FLAG field, calls the appropriate opcodes to delete associated session and resource information:

- **1** = Delete only the **/reservation_active** object.
- **2** = Delete only the **/active_session** object (or a subclass thereof).
- **3** = Delete both objects.

See the discussion on how BRM cancels prepaid service authorization in *BRM Telco Integration*.

PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD

Reads prepaid service authorization and reauthorization threshold information from the **/config/auth_reauth_info** object.

See the discussion son how BRM reduces authorization latencies and how BRM uses a scaled delay time to reduce network spikes during a tariff change in *BRM Telco Integration*.

Note: This opcode is used for light-weight authorization. See the discussion on using light-weight authorization in *BRM Telco Integration*.

Return Values

This opcode returns the following for a particular service for a particular account on the output flist:

- The traffic-light status with possible values of PIN_RESOURCE_STATUS_GREEN, PIN_RESOURCE_STATUS_YELLOW, and PIN_RESOURCE_STATUS_RED
- The scaled delay time
- The reservation amount

PCM_OP_ACT_END_SESSION

For prepaid sessions, closes active sessions, releases their resources, gathers their information to store in the BRM database, deletes them from IMDB Cache, and ensures that they are rated and that their account balances are updated.

For postpaid sessions, logs the end of a session event and assesses any charges.

See the following discussions:

- How BRM ends prepaid sessions in *BRM Telco Integration*
- Rating and recording session events in *BRM Setting Up Pricing and Rating*

PCM_OP_ACT_FIND

Locates a customer's account by using the login name.

See the discussion on finding a customer's account information in *BRM Managing Customers*.

Important: 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.

PCM_OP_ACT_FIND_VERIFY

Authenticates a user action.

This opcode 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.

> **Note:** These checks are applicable only to **/service/pcm_client** and **/service/admin_client**.

See the discussions on authenticating customers by using your custom application in *BRM Managing Customers* and configuring the maximum number of invalid login attempts in *BRM System Administrator's Guide*.

PCM_OP_ACT_GET_CANDIDATE_RUMS

Creates a candidate ratable usage metric (RUM) by combining data from the input flist and the **/config/rum** object.

See the discussion on generating ratable usage metrics (RUMs) in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_HANDLE_OOD_EVENT

Calls PCM_OP_RERATE_INSERT_RERATE_REQUEST to create a rerate job.

This opcode is called by event notification when the **/event/notification/activity/out_of_order** event occurs.

See the discussion on how BRM rerates out-of-order events in *BRM Configuring and Running Billing*.

PCM_OP_ACT_LOAD_SESSION

Creates, rates, and records a session event as a single operation. This opcode is valuable as a tool to load sessions in real time.

See the discussion on creating opcodes for loading event data in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_LOG_USER_ACTIVITY

Logs user activities in the **/user_activity** storable class.

See the discussion about logging CSR activities in *BRM System Administrator's Guide*.

PCM_OP_ACT_LOGIN

Authorizes users to start login sessions.

This opcode also stores the PIN_FLD_NAP_IP_ADDRESS in the **/event/session** object on successful login. This information is used for logging CSR activities.

See the discussions on starting sessions in *BRM Setting Up Pricing and Rating* and logging customer service representative activities in *BRM System Administrator's Guide*.

PCM_OP_ACT_LOGOUT

Records the end of a login session.

See the discussion on recording the end of a session in *BRM Setting Up Pricing and Rating*.

PCM_OP_ACT_MULTI_AUTHORIZE

Authorizes or rates multiple prepaid services with a single call.

This opcode takes an array of services and a mode as input. Depending on the mode, it does the following:

- Authorizes and rates each service by making a call to PCM_OP_ACT_AUTHORIZE for each service.
- Performs calc-only rating by making a call to PCM_OP_ACT_USAGE for each service.

These actions are all performed in a single opcode call. All services passed in are authorized or rated individually in that call. In all instances, this opcode returns a results array containing the result of each individual transaction:

- If the transaction was successful, the results array element contains a balance array with the balance impact for each resource affected and a quantity authorized for each service.
- If the transaction failed, the results array element contains the reasons for that failure.

The actions of this opcode are controlled by the mode. These are the options:

- **Mode0:** Performs authorization and resource reservation and then rates each service passed in. Use this mode if your BRM implementation maintains balances in the BRM database. This opcode makes a call to PCM_OP_ACT_AUTHORIZE for each of the services passed in. PCM_OP_ACT_AUTHORIZE then calls PCM_OP_ACT_USAGE to reserve the resources and impact both monetary and nonmonetary balances in the BRM database.
- **Mode1:** Performs calc-only rating for requested balances for each service passed in. Using available balances in the BRM database, this opcode calculates whether the account has sufficient resources for the requested service and returns the result. Use this mode if you maintain balances in the BRM database but do not want to reserve those balances. This opcode calls PCM_OP_ACT_USAGE to perform the rating. This mode does not reserve resources.
- **Mode2:** Performs calc-only rating for requested balances. This opcode calculates whether the account has sufficient resources for the requested service and returns the result. It uses different available balances for the monetary and nonmonetary resource calculations:
  - For nonmonetary resources, this opcode makes the calculation using the available balances in the BRM database.
  - For monetary resources, this opcode makes the calculation using the available balances passed in.

    Important: The monetary resources passed in using Mode2 take priority over those in the BRM database. If a monetary resource is passed in, and a corresponding resource exists in the BRM database, the resource passed in is used.

  Use this mode if you maintain monetary balances in a non-BRM database and nonmonetary balances in the BRM database. This opcode calls PCM_OP_ACT_USAGE to perform the rating. This mode does not reserve resources.

Example 1–1 Mode0 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_MODE           ENUM [0] 0
0 PIN_FLD_SERVICES      ARRAY [0] allocated 20, used 4
1     PIN_FLD_LOGIN           STR [0] "login"
1     PIN_FLD_MIN_QUANTITY DECIMAL [0] 0
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 2
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID    STR [0] "0044191009855-11126216443"
2         PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
3             PIN_FLD_NUMBER_OF_UNITS    INT [0] 0
3             PIN_FLD_DIRECTION      ENUM [0] 0
1     PIN_FLD_EVENT        SUBSTRUCT [0] allocated 20, used 10
2         PIN_FLD_POID           POID [0] 0.0.0.1 /event/session/telco/gsm -1 0
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/telco/gsm/telephony
70215 0
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 72071 0
2         PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
2         PIN_FLD_ACTIVE_SESSION_ID    STR [0] "0044191009855-11126216443"
2         PIN_FLD_USAGE_TYPE      STR [0] "NAT"
2         PIN_FLD_START_T      TSTAMP [0] (1116918240) Tue May 24 00:04:00 2005
2         PIN_FLD_END_T        TSTAMP [0] (1116918840) Tue May 24 00:14:00 2005
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID    STR [0] "0044191009855-1112621644"
2         PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 4
3             PIN_FLD_NUMBER_OF_UNITS    INT [0] 0
3             PIN_FLD_DIRECTION      ENUM [0] 0
3             PIN_FLD_BYTES_IN        INT [0] 0
3             PIN_FLD_BYTES_OUT       INT [0] 0
1 PIN_FLD_RESERVATION_OBJ   POID [0] 0.0.0.1 /reservation/active -1 0
```

Example 1–2 Mode0 Sample Output Flist (Successful)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 10
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 72071 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm 212136 0
1     PIN_FLD_AUTHORIZATION_ID    STR [0] "0044191009855-11126216443"
1     PIN_FLD_RESULT         ENUM [0] 1
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/telco/gsm/telephony 70215 0
1     PIN_FLD_QUANTITY     DECIMAL [0] 600
1     PIN_FLD_RATING_STATUS   ENUM [0] 0
1     PIN_FLD_RESERVATION_OBJ   POID [0] 0.0.0.1 /reservation 209576 0
1     PIN_FLD_EXPIRATION_T TSTAMP [0] (1117159330) Thu May 26 19:02:10 2005
1     PIN_FLD_BALANCES      ARRAY [978] allocated 20, used 2?
2         PIN_FLD_AMOUNT       DECIMAL [0] -100.000
2         PIN_FLD_AVAILABLE_RESOURCE_LIMIT DECIMAL [0] 800.000
```

Example 1–3 Mode0 Sample Output Flist (Unsuccessful)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 2
1     PIN_FLD_RESULT         ENUM [0] 0
1     PIN_FLD_REASON         ENUM [0] 8
```

Example 1–4 Mode1 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_MODE           ENUM [0] 1
0 PIN_FLD_SERVICES      ARRAY [0] allocated 20, used 4
```

```
1     PIN_FLD_LOGIN           STR [0] "login"
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 2
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID    STR [0] "0044191009855-1112621644"
2         PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
3             PIN_FLD_NUMBER_OF_UNITS    INT [0] 0
3             PIN_FLD_DIRECTION      ENUM [0] 0
1     PIN_FLD_EVENT        SUBSTRUCT [0] allocated 20, used 10
2         PIN_FLD_POID           POID [0] 0.0.0.1 /event/session/telco/gsm -1 0
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/telco/gsm/telephony
70215 0
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 72071 0
2         PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
2         PIN_FLD_ACTIVE_SESSION_ID    STR [0] "0044191009855-1112621644"
2         PIN_FLD_USAGE_TYPE      STR [0] "NAT"
2         PIN_FLD_START_T      TSTAMP [0] (1116918240) Tue May 24 00:04:00 2005
2         PIN_FLD_END_T        TSTAMP [0] (1116918840) Tue May 24 00:14:00 2005
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID    STR [0] "0044191009855-1112621644"
2         PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 4
3             PIN_FLD_NUMBER_OF_UNITS    INT [0] 0
3             PIN_FLD_DIRECTION      ENUM [0] 0
3             PIN_FLD_BYTES_IN        INT [0] 0
3             PIN_FLD_BYTES_OUT       INT [0] 0
```

Example 1–5 Mode1 Sample Output Flist (Successful)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_RESULTS       ARRAY [1] allocated 20, used 10
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/session/telco/gsm -1 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 72071 0
1     PIN_FLD_RESULT         ENUM [0] 1
1     PIN_FLD_RATING_STATUS   ENUM [0] 0
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/telco/gsm/telephony 70215 0
1     PIN_FLD_NET_QUANTITY DECIMAL [0] 600
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
1     PIN_FLD_RUM_NAME        STR [0] "Duration"
1     PIN_FLD_UNIT           ENUM [0] 1
1     PIN_FLD_BALANCES      ARRAY [978] allocated 20, used 1
2         PIN_FLD_AMOUNT       DECIMAL [0] -121.200
```

Example 1–6 Mode2 Sample Input Flist

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_MODE ENUM [0] 2
0 PIN_FLD_SERVICES ARRAY [0] allocated 20, used 5
1     PIN_FLD_LOGIN STR [0] "login"
1     PIN_FLD_MIN_QUANTITY DECIMAL [0] 0
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 2
2         PIN_FLD_TELCO_INFO SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID STR [0] "0044191009855-1112621644"
2         PIN_FLD_GSM_INFO SUBSTRUCT [0] allocated 20, used 2
3             PIN_FLD_NUMBER_OF_UNITS INT [0] 0
3             PIN_FLD_DIRECTION ENUM [0] 0
1     PIN_FLD_EVENT SUBSTRUCT [0] allocated 20, used 10
2         PIN_FLD_POID POID [0] 0.0.0.1 /event/session/telco/gsm -1 0
2         PIN_FLD_SERVICE_OBJ POID [0] 0.0.0.1 /service/telco/gsm/telephony 70215
0
2         PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 72071 0
2         PIN_FLD_PROGRAM_NAME STR [0] "testnap"
2         PIN_FLD_ACTIVE_SESSION_ID STR [0] "0044191009855-1112621644"
```

```
2         PIN_FLD_USAGE_TYPE STR [0] "NAT"
2         PIN_FLD_START_T TSTAMP [0] (1116918240) Tue May 24 00:04:00 2005
2         PIN_FLD_END_T TSTAMP [0] (1116918840) Tue May 24 00:14:00 2005
2         PIN_FLD_TELCO_INFO SUBSTRUCT [0] allocated 20, used 1
3             PIN_FLD_NETWORK_SESSION_ID STR [0] "0044191009855-1112621644"
2         PIN_FLD_GSM_INFO SUBSTRUCT [0] allocated 20, used 4
3             PIN_FLD_NUMBER_OF_UNITS INT [0] 0
3             PIN_FLD_DIRECTION ENUM [0] 0
3             PIN_FLD_BYTES_IN INT [0] 0
3             PIN_FLD_BYTES_OUT INT [0] 0
1     PIN_FLD_BALANCES ARRAY [978] allocated 20, used 6
2         PIN_FLD_RESERVED_AMOUNT DECIMAL [0] -400
2         PIN_FLD_NEXT_BAL DECIMAL [0] 0
2         PIN_FLD_CURRENT_BAL DECIMAL [0] 39.95
2         PIN_FLD_CREDIT_LIMIT DECIMAL [0] 0
2         PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
2         PIN_FLD_CREDIT_THRESHOLDS INT [0] 0
```

Example 1–7 Mode2 Sample Output Flist (successful)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 10
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/session/telco/gsm -1 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 72071 0
1     PIN_FLD_RESULT         ENUM [0] 1
1     PIN_FLD_RATING_STATUS   ENUM [0] 0
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/telco/gsm/telephony 70215 0
1     PIN_FLD_NET_QUANTITY DECIMAL [0] 600
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
1     PIN_FLD_RUM_NAME        STR [0] "Duration"
1     PIN_FLD_UNIT           ENUM [0] 1
1     PIN_FLD_BALANCES      ARRAY [978] allocated 20, used 1
2         PIN_FLD_AMOUNT       DECIMAL [0] -121.200
```

See the discussion on authorizing multiple services for a user with a single call in *BRM Telco Integration*.

Return Values

This opcode always returns the PIN_FLD_RESULTS array on the output flist. If the opcode call was unsuccessful, this opcode also returns a numeric value in the PIN_FLD_REASON field to provide as much useful information as possible.

Because this opcode calls PCM_OP_ACT_AUTHORIZE, the numerical codes returned in PIN_FLD_REASON may come from either PCM_OP_ACT_AUTHORIZE or PCM_OP_ACT_MULTI_AUTHORIZE:

- Error codes 1 through 100 come from PCM_OP_ACT_AUTHORIZE and are defined in the **pin_reserve.h** file.
- Error codes 101 and higher come from PCM_OP_ACT_MULTI_AUTHORIZE and are defined in the **pin_act.h** file.

PCM_OP_ACT_REAUTHORIZE

Reauthorizes prepaid services.

If PIN_FLD_PIGGYBACK_FLAG is enabled in the input flist, this opcode returns information on the credit thresholds breached in the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist.

See the following discussions:

- How BRM reauthorizes prepaid services in *BRM Telco Integration*
- Providing credit threshold breach information in in-session notifications for network connectivity applications in *BRM Telco Integration*

PCM_OP_ACT_SCHEDULE_CREATE

Creates a **/schedule** object, which defers a single action to a predetermined date and time.

See the discussion on scheduling deferred actions in *BRM Managing Customers*.

PCM_OP_ACT_SCHEDULE_DELETE

Deletes **/schedule** objects.

See the discussion on deleting deferred actions in *BRM Managing Customers*.

PCM_OP_ACT_SCHEDULE_EXECUTE

Executes any deferred actions in specified **/schedule** objects.

This opcode is called directly by the **pin_deferred_act** billing utility.

See the discussion on executing deferred actions in *BRM Managing Customers*.

PCM_OP_ACT_SCHEDULE_MODIFY

Modifies the text description of an existing **/schedule** object.

See the discussion on modifying deferred action descriptions in *BRM Managing Customers*.

PCM_OP_ACT_START_SESSION

Records the start of a prepaid or postpaid session event.

See the following discussions:

- (Prepaid) How BRM starts prepaid sessions in *BRM Telco Integration*
- (Postpaid) Recording the start of a session in *BRM Setting Up Pricing and Rating*

PCM_OP_ACT_UPDATE_SESSION

During prepaid and postpaid sessions, updates session event information.

At the end of prepaid and postpaid sessions, records the session end time.

See the following discussions:

- (Prepaid) How BRM updates prepaid sessions in *BRM Telco Integration*
- (Prepaid) How BRM updates and reauthorizes prepaid sessions in *BRM Telco Integration*
- (Postpaid) Updating a session event in *BRM Setting Up Pricing and Rating*

PCM_OP_ACT_USAGE

Rates and records an event.

If balance monitoring is enabled, this opcode retrieves a list of balance monitors.

If event notification is enabled, checks whether the event is in your system's event notification list. If it is, calls the associated opcodes.

If PCM_OP_ACT_USAGE is called with PIN_FLD_PIGGYBACK_FLAG set to **1** (enabled), the opcode calculates the credit threshold breaches and returns any breach information in its output flist.

See the following discussions:

- How BRM rates and records usage events in *BRM Setting Up Pricing and Rating*
- Balance monitoring in *BRM Managing Accounts Receivable*
- Using event notification in *BRM Developer's Guide*
- How BRM creates credit threshold breach notifications in *BRM Telco Integration*

Account Dump FM Policy Opcodes

Use the opcodes in Table 1–4 to customize Account Dump Utility (ADU) validation and output file format.

Header File

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

Opcode Index

Table 1–4 Account Dump FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ADU_POL_DUMP	A policy hook that allows you to convert the ADU output format to a format other than XML.	Recommended
PCM_OP_ADU_POL_VALIDATE	A policy hook that allows you to define custom validations for validating account data.	Recommended

PCM_OP_ADU_POL_DUMP

This policy opcode is a hook and is called by the PCM_OP_ADU_VALIDATE opcode.

By default, PCM_OP_ADU_VALIDATE dumps account information in XML format. The PCM_OP_ADU_POL_DUMP policy opcode allows you to convert the account information in the input flist into a format other than XML; for example, CSV.

Example 1–8 Sample Input Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /account 86394 0
0 PIN_FLD_ACCOUNT                   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_OBJ_ELEM              ARRAY [0] allocated 20, used 2
2         PIN_FLD_POID              POID [0] 0.0.0.1 /account 86394 10
2         PIN_FLD_NAMEINFO          ARRAY [1] allocated 20, used 19
3             PIN_FLD_ADDRESS       STR [0] "accnt_1"
3             PIN_FLD_CANON_COMPANY STR [0] ""
3             PIN_FLD_CANON_COUNTRY STR [0] "IN"
3             PIN_FLD_CITY          STR [0] "accnt_1"
3             PIN_FLD_COMPANY       STR [0] ""
3             PIN_FLD_CONTACT_TYPE  STR [0] "Account holder"
3             PIN_FLD_COUNTRY       STR [0] "IN"
3             PIN_FLD_EMAIL_ADDR    STR [0] "accnt_1"
3             PIN_FLD_FIRST_CANON   STR [0] "accnt_1"
3             PIN_FLD_FIRST_NAME    STR [0] "accnt_1"
3             PIN_FLD_LAST_CANON    STR [0] "accnt_1"
3             PIN_FLD_LAST_NAME     STR [0] "accnt_1"
3             PIN_FLD_MIDDLE_CANON  STR [0] "accnt_1"
3             PIN_FLD_MIDDLE_NAME   STR [0] "accnt_1"
3             PIN_FLD_SALUTATION    STR [0] ""
3             PIN_FLD_SERVICE_OBJ   POID [0] 0.0.0.0  0 0
3             PIN_FLD_STATE         STR [0] "accnt_1"
3             PIN_FLD_TITLE         STR [0] ""
3              PIN_FLD_ZIP           STR [0] "11111"
0 PIN_FLD_SERVICES                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_DEVICES                    ARRAY [0] allocated 0, used 0
0 PIN_FLD_BILLINFO                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_BILLS                      ARRAY [0] allocated 0, used 0
0 PIN_FLD_ITEMS                      ARRAY [0] allocated 0, used 0
0 PIN_FLD_EVENTS                     ARRAY [0] allocated 0, used 0
0 PIN_FLD_PAYINFO                    ARRAY [0] allocated 0, used 0
0 PIN_FLD_PROFILES                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_GROUPS                     ARRAY [0] allocated 0, used 0
0 PIN_FLD_BALANCE_GROUPS             ARRAY [0] allocated 0, used 0
0 PIN_FLD_CUSTOM_INFO                   SUBSTRUCT [0] allocated 0, used 0
```

Example 1–9 Sample Output Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /account 4263658 0
0 PIN_FLD_RESULTS       ARRAY [0]
1 PIN_FLD_BUFFER          BUF [0] ""
```

PCM_OP_ADU_POL_VALIDATE

This policy opcode is a hook and is called by the PCM_OP_ADU_VALIDATE opcode.

By default, PCM_OP_ADU_VALIDATE performs the predefined validations enabled in the ADU **pin.conf** file. The PCM_OP_ADU_POL_VALIDATE policy opcode allows you to define custom validations for validating the account information provided in the input flist.

Note: 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**. See the discussion on account data validation in *BRM Managing Customers*.

Example 1–10 Sample Input Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /account 86394 0
0 PIN_FLD_ACCOUNT                   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_OBJ_ELEM              ARRAY [0] allocated 20, used 2
2         PIN_FLD_POID              POID [0] 0.0.0.1 /account 86394 10
2         PIN_FLD_NAMEINFO          ARRAY [1] allocated 20, used 19
3             PIN_FLD_ADDRESS       STR [0] "accnt_1"
3             PIN_FLD_CANON_COMPANY STR [0] ""
3             PIN_FLD_CANON_COUNTRY STR [0] "IN"
3             PIN_FLD_CITY          STR [0] "accnt_1"
3             PIN_FLD_COMPANY       STR [0] ""
3             PIN_FLD_CONTACT_TYPE  STR [0] "Account holder"
3             PIN_FLD_COUNTRY       STR [0] "IN"
3             PIN_FLD_EMAIL_ADDR    STR [0] "accnt_1"
3             PIN_FLD_FIRST_CANON   STR [0] "accnt_1"
3             PIN_FLD_FIRST_NAME    STR [0] "accnt_1"
3             PIN_FLD_LAST_CANON    STR [0] "accnt_1"
3             PIN_FLD_LAST_NAME     STR [0] "accnt_1"
3             PIN_FLD_MIDDLE_CANON  STR [0] "accnt_1"
3             PIN_FLD_MIDDLE_NAME   STR [0] "accnt_1"
3             PIN_FLD_SALUTATION    STR [0] ""
3             PIN_FLD_SERVICE_OBJ   POID [0] 0.0.0.0  0 0
3             PIN_FLD_STATE         STR [0] "accnt_1"
3             PIN_FLD_TITLE         STR [0] ""
3              PIN_FLD_ZIP           STR [0] "11111"
0 PIN_FLD_SERVICES                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_DEVICES                    ARRAY [0] allocated 0, used 0
0 PIN_FLD_BILLINFO                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_BILLS                      ARRAY [0] allocated 0, used 0
0 PIN_FLD_ITEMS                      ARRAY [0] allocated 0, used 0
0 PIN_FLD_EVENTS                     ARRAY [0] allocated 0, used 0
0 PIN_FLD_PAYINFO                    ARRAY [0] allocated 0, used 0
0 PIN_FLD_PROFILES                   ARRAY [0] allocated 0, used 0
0 PIN_FLD_GROUPS                     ARRAY [0] allocated 0, used 0
0 PIN_FLD_BALANCE_GROUPS             ARRAY [0] allocated 0, used 0
0 PIN_FLD_CUSTOM_INFO                   SUBSTRUCT [0] allocated 0, used 0
```

Example 1–11 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 4263658 0
0 PIN_FLD_RESULTS           ARRAY [0]
1 PIN_FLD_NAME                STR [0] "struct_valid_01"
```

Account Dump FM Standard Opcodes

The opcode in Table 1–5 performs Account Dump Utility (ADU) functions, which are used to validate account information.

Header File

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

Opcode Index

Table 1–5 Account Dump FM Standard Opcodes

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

PCM_OP_ADU_VALIDATE

This opcode performs two main functions:

- When the **-dump** option is specified, this opcode searches the BRM database for the list of objects specified in the ADU configuration file (**pin.conf**) and dumps their contents into the ADU output file.
- When the **-validate** option is specified, this opcode validates the object contents by performing the validations enabled in the ADU **pin.conf** and the custom validations defined in the PCM_OP_ADU_POL_VALIDATE policy opcode.

Example 1–12 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /account 4109485054 0
0 PIN_FLD_DIRECTORY               STR [0] "/opt/portal/7.3.1/sys/diagnostics/pin_
adu_validate/out"
0 PIN_FLD_OBJECTCLASS             STR [0]"/account, /au_account, /service/email"
0 PIN_FLD_FIELD_NAME                 STR [0] "/account:PIN_FLD_POID, PIN_FLD_
NAMEINFO; /service/email:PIN_FLD_POID, PIN_FLD_SERVICE_EMAIL.PIN_FLD_PATH;"
0 PIN_FLD_EXT_INFO_STR            STR [0] ".xml"
0 PIN_FLD_START_T                 TSTAMP [0] (1175385600)
0 PIN_FLD_END_T                   TSTAMP [0] (1177977600)
0 PIN_FLD_PARAMS                  ARRAY [0]
1   PIN_FLD_PARAM_NAME            STR [0] "-dump"
0 PIN_FLD_PARAMS                  ARRAY [1]
1   PIN_FLD_PARAM_NAME            STR [0] "-validate"
0 PIN_FLD_VALIDATION_SPECIFICS    ARRAY [0]
1   PIN_FLD_SCENARIO_NAME         STR [0] "struct_valid_01"
0 PIN_FLD_VALIDATION_SPECIFICS    ARRAY [1]
1   PIN_FLD_SCENARIO_NAME         STR [0] "struct_valid_02"
0 PIN_FLD_VALIDATION_SPECIFICS    ARRAY [2]
1   PIN_FLD_SCENARIO_NAME            STR [0] "dynamic_valid_01"
```

Example 1–13 Sample Output Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /account 4109485054 0
```

Accounts Receivable FM Policy Opcodes

The opcodes in Table 1–6 manage accounts receivable (A/R) functions such as adjustments, disputes, and write-offs.

Header File

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

Opcode Index

Table 1–6 Accounts Receivable FM Policy Opcodes

Opcode	Description	Use
PCM_OP_AR_POL_PRE_EVENT_ADJUSTMENT	Allows you to customize the input flist by adding or deleting events.	Recommended
PCM_OP_AR_POL_REVERSE_WRITEOFF	Retrieves the list of write-off items that need to be reversed from the **/profile/writeoff** object. See the discussion on customizing write-off reversals in *BRM Managing Accounts Receivable*.	Recommended

PCM_OP_AR_POL_PRE_EVENT_ADJUSTMENT

Allows 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

Retrieves the write-off reversal items from the **/profile/writeoff** object. During the write-off reversal process, this policy opcode is called to supply a list of write-off items that require reversal if that list is not provided by any other means.

This opcode is called by the PCM_OP_AR_REVERSE_WRITEOFF standard opcode.

See the discussion on customizing write-off reversals in *BRM Managing Accounts Receivable*.

Example 1–14 Sample Input Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 11857 1
0 PIN_FLD_PROGRAM_NAME       STR [0] "DSG-reversal"
```

Example 1–15 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 11857 1
0 PIN_FLD_PROGRAM_NAME       STR [0] "DSG-reversal"
0 PIN_FLD_REVERSALS        ARRAY [0] allocated 20, used 1
1     PIN_FLD_ITEM_OBJ      POID [0] 0.0.0.1 /item/writeoff 9561 0
0 PIN_FLD_REVERSALS        ARRAY [1] allocated 20, used 1
1     PIN_FLD_ITEM_OBJ      POID [0] 0.0.0.1 /item/writeoff 9834 0
```

Accounts Receivable FM Standard Opcodes

The accounts receivable opcodes in Table 1–7 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.

Header File

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

Opcode Index

Table 1–7 Accounts Receivable FM Standard Opcodes

Opcode	Description	Use
PCM_OP_AR_ACCOUNT_ADJUSTMENT	Debits or credits an account balance for currency adjustments. See the discussions on adjusting accounts, subscription services, and member services in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_ACCOUNT_WRITEOFF	Performs write-off adjustments for an account or account hierarchy with or without tax implication. See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_BILL_ADJUSTMENT	Credits the currency balance of a bill. See the discussion on adjusting bills in *BRM Managing Accounts Receivable*.	Recommended
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	Recommended
PCM_OP_AR_BILL_DISPUTE	Opens a dispute against a bill. See the discussion on disputing bills in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_BILL_SETTLEMENT	Settles a dispute opened against a bill. See the discussion on settling disputed bills in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_BILL_WRITEOFF	Performs write-off adjustments against a specific bill. See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_BILLINFO_WRITEOFF	Performs write-off adjustments against a specific bill unit	Recommended

Table 1–7 (Cont.) Accounts Receivable FM Standard Opcodes

Opcode	Description	Use
PCM_OP_AR_EVENT_ADJUSTMENT	Adjusts balance impact of an event on an account's balance. See the discussion on adjusting events in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_EVENT_DISPUTE	Opens a dispute against one or more events in an account. See the discussion on disputing events in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_EVENT_SETTLEMENT	Settles one or more dispute events in a dispute item. See the discussion on settling disputed events in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_ACCT_ACTION_ITEMS	Retrieves the list of A/R items applied to all bill units in an account or to a single bill unit. See the discussion on retrieving A/R items that apply to a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_ACCT_BAL_SUMMARY	Retrieves the consolidated balances of the applied amount, unapplied amount, open bill due amount, pending bill due amount, and total dispute amount for all bill units in an account. See the discussion on retrieving a balance summary in *BRM Managing Accounts Receivable*.	Recommended
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 opcode performs the search based on start and end times. See the discussion on retrieving a list of bills for a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_ACTION_ITEMS	Retrieves the list of A/R items applied to a bill unit (**/billinfo** object). See the discussion on retrieving A/R items that apply to a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_BAL_SUMMARY	Retrieves the applied, unapplied, open bill due, pending bill due, and total dispute balances for a given bill unit (**/billinfo** object). See the discussion on retrieving a balance summary in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_BILL_ITEMS	Retrieves the list of bill items for a bill unit (**/billinfo** object). See the discussion on retrieving a list of bill items for a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_BILLS	Retrieves a list of bills for a bill unit (**/billinfo** object). See the discussion on retrieving a list of bills for a bill unit in *BRM Managing Accounts Receivable*.	Recommended

Table 1–7 (Cont.) Accounts Receivable FM Standard Opcodes

Opcode	Description	Use
PCM_OP_AR_GET_DISPUTE_DETAILS	Retrieves all event-level and item-level disputes and the aggregated amount of each resource for the dispute events associated with an item dispute. See the discussion on retrieving dispute details for a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_DISPUTES	Retrieves details of all disputed bill items for a given bill unit (**/billinfo** object). See the discussion on retrieving dispute details for a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_ITEM_DETAIL	Retrieves details for a specified A/R action or bill item. These details are for currency resources. See the discussion on retrieving details about a specific A/R item or bill item in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_GET_ITEMS	Retrieves details for a specified A/R action or bill item. These details are for currency and noncurrency resources. See the discussion on retrieving details about a specific A/R item or bill item in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_ITEM_ADJUSTMENT	Makes adjustments against an item. See the discussion on adjusting items in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_ITEM_DISPUTE	Files a dispute against an item. See the discussion on disputing items in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_ITEM_SETTLEMENT	Settles an item that is in dispute. See the discussion on settling disputed items in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_AR_ITEM_WRITEOFF	Makes write-off adjustments against an item. See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.	Recommended

Table 1–7 (Cont.) Accounts Receivable FM Standard Opcodes

Opcode	Description	Use
PCM_OP_AR_RESOURCE_AGGREGATION	Calculates the aggregated amount of each resource for an event. See the discussion on retrieving details on available resources in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_AR_REVERSE_REFUND	Reverses a SEPA Credit Transfer refund that was previously recorded in BRM. See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_AR_REVERSE_WRITEOFF	Reverses write-offs on written-off bills, written-off bill units, and written-off bill items when a payment is received. See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.	Limited

PCM_OP_AR_ACCOUNT_ADJUSTMENT

Debits or credits an account balance for currency adjustments. This opcode is called by BRM client applications to adjust the balance impacts of an event.

See the discussion on adjusting accounts, subscription services, and member services in *BRM Managing Accounts Receivable*.

Example 1–16 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 26376 13
0 PIN_FLD_AMOUNT        DECIMAL [0] -1
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_CURRENCY          INT [0] 840
0 PIN_FLD_STR_VERSION       INT [0] 8
0 PIN_FLD_STRING_ID         INT [0] 3
0 PIN_FLD_DESCR             STR [0] "Account debited by mistake"
0 PIN_FLD_BAL_GRP_OBJ      POID [0] 0.0.0.1 /balance_group 27272 0
0 PIN_FLD_ITEM_NO            STR [0] "A1-58"
0 PIN_FLD_FLAGS             INT [0] 2
```

Example 1–17 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /account 26376 13
0 PIN_FLD_RESULTS          ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/adjustment/account
                                     220025470857538664 0
```

PCM_OP_AR_ACCOUNT_WRITEOFF

Performs write-off of one or more A/R bill units for an account. 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.

See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.

Example 1–18 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /bill 198434 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "Customer Center"
0 PIN_FLD_DESCR           STR [0] "abce"
0 PIN_FLD_FLAGS           INT [0] 1
```

Example 1–19 Sample Output Flist

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /account 198434 0
0 PIN_FLD_RESULT       ENUM [0] 1
0 PIN_FLD_RESULTS     ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID     POID [0] 0.0.0.1 /event/billing/writeoff/account
216964430485979690 0
```

PCM_OP_AR_BILL_ADJUSTMENT

Credits the currency balance of an account's AR bill. This opcode is called by BRM client applications to adjust the balance impacts of items associated with the specified bill.

See the discussion on adjusting bills in *BRM Managing Accounts Receivable*.

Example 1–20 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /bill 27012 0
0 PIN_FLD_AMOUNT        DECIMAL [0] -2
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_REASON_DOMAIN_ID  INT [0] 16
0 PIN_FLD_REASON_ID         INT [0] 2
0 PIN_FLD_DESCR             STR [0] "Customer unaware of charges"
0 PIN_FLD_FLAGS             INT [0] 2
```

Example 1–21 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /bill 27012 0
0 PIN_FLD_RESULTS          ARRAY [0] allocated 2, used 2
1     PIN_FLD_POID          POID [0] 0.0.0.1 /item/misc 26628 15
1     PIN_FLD_EVENT_OBJ     POID [0] 0.0.0.1 /event/billing/adjustment/item
                                     220025470857537860 0
0 PIN_FLD_RESULT            ENUM [0] 1
0 PIN_FLD_DESCR              STR [0] "Successful"
```

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 is called by Customer Center when a CSR allocates credit amounts to bills that have a positive balance. The opcode takes as input the **/bill** object and corresponding **/billinfo** object of both the source bill and destination bill or bills.

See the discussion on adjusting bills in *BRM Managing Accounts Receivable*.

PCM_OP_AR_BILL_DISPUTE

Opens a dispute against the account's A/R bill. This opcode is called by BRM client applications to dispute the items associated with the specified bill.

See the discussion on disputing bills in *BRM Managing Accounts Receivable*.

Example 1–22 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /bill 20529 2
0 PIN_FLD_AMOUNT        DECIMAL [0] -11.23
0 PIN_FLD_CURRENCY          INT [0] 840
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DESCR             STR [0] "Customer unhappy with charges"
```

Example 1–23 Sample Output Flist

```
0 PIN_FLD_POI              POID [0] 0.0.0.1 /bill 20529 2
0 PIN_FLD_RESULTS         ARRAY [0] allocated 2, used 2
1     PIN_FLD_POID         POID [0] 0.0.0.1 /item/cycle_forward 23063 1
1     PIN_FLD_EVENT_OBJ    POID [0] 0.0.0.1 /event/billing/dispute/item
220025470857537060 0
0 PIN_FLD_RESULT           ENUM [0] 1
0 PIN_FLD_DESCR             STR [0] "Successful"
```

PCM_OP_AR_BILL_SETTLEMENT

Settles an A/R bill that is in dispute. This opcode is called by BRM client applications to settle the disputed items associated with the specified bill.

See the discussion on settling disputed bills in *BRM Managing Accounts Receivable*.

Example 1–24 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /bill 15436 0
0 PIN_FLD_AMOUNT        DECIMAL [0] -3
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DESCR             STR [0] "Bill settled for full amount"
```

Example 1–25 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /bill 15436 0
0 PIN_FLD_RESULTS         ARRAY [0] allocated 20, used 2
1     PIN_FLD_POID         POID [0] 0.0.0.1 /item/misc 13836 7
1     PIN_FLD_EVENT_OBJ    POID [0] 0.0.0.1 /event/billing/settlement/item
                                    219743995880815580 0
0 PIN_FLD_RESULTS         ARRAY [1] allocated 20, used 2
1     PIN_FLD_POID         POID [0] 0.0.0.1 /item/misc 14220 5
1     PIN_FLD_EVENT_OBJ    POID [0] 0.0.0.1 /event/billing/settlement/item
                                    219743995880812604 0
0 PIN_FLD_RESULT           ENUM [0] 1
0 PIN_FLD_DESCR             STR [0] "Successful"
```

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.

See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.

Example 1–26 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /bill 198418 8
0 PIN_FLD_PROGRAM_NAME    STR [0] "Customer Center"
0 PIN_FLD_DESCR           STR [0] "abce"
0 PIN_FLD_FLAGS           INT [0] 4
```

Example 1–27 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /bill 198418 8
0 PIN_FLD_RESULT         ENUM [0] 1
0 PIN_FLD_RESULTS       ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID       POID [0] 0.0.0.1 /event/billing/writeoff/bill
216964430485981226 0
```

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.

See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.

To write-off *all* bill units of an account, see "PCM_OP_AR_ACCOUNT_WRITEOFF".

Example 1–28 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 143727 8
0 PIN_FLD_PROGRAM_NAME    STR [0] "Customer Center"
0 PIN_FLD_DESCR           STR [0] "Customer not happy"
0 PIN_FLD_FLAGS           INT [0] 4
```

Example 1–29 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 143727 8
0 PIN_FLD_RESULT         ENUM [0] 1
0 PIN_FLD_RESULTS       ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID       POID [0] 0.0.0.1 /event/billing/writeoff/billinfo
216964430485981226 0
```

PCM_OP_AR_EVENT_ADJUSTMENT

Adjusts the balance impact of an event on an account's balance. This opcode receives a list of events to be adjusted as a batch on the input flist, and returns the adjusted data on the output flist.

See the discussion on adjusting events in *BRM Managing Accounts Receivable*.

Example 1–30 Sample Input Flist

```
0 PIN_FLD_EVENTS             ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID            POID [0] 0.0.0.1 /event/session 220025470857535512 0
0 PIN_FLD_POID                POID [0] 0.0.0.1 /account 26216 13
0 PIN_FLD_ADJUSTMENT_INFO    ARRAY [0] allocated 2, used 2
1     PIN_FLD_PERCENT      DECIMAL [0] 25
1     PIN_FLD_RESOURCE_ID      INT [0] 840
0 PIN_FLD_PROGRAM_NAME         STR [0] "Customer Center"
0 PIN_FLD_DESCR                STR [0] "Service down"
0 PIN_FLD_REASON_DOMAIN_ID     INT [0] 14
0 PIN_FLD_REASON_ID            INT [0] 1
```

Example 1–31 Sample Output Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 26216 13
0 PIN_FLD_RESULT                        ENUM [0] 1
0 PIN_FLD_DESCR                          STR [0] "Successful"
0 PIN_FLD_ADJUSTMENT_INFO              ARRAY [0] allocated 20, used 5
1     PIN_FLD_RESOURCE_ID                INT [0] 840
1     PIN_FLD_AMOUNT                 DECIMAL [0] 1
1     PIN_FLD_AMOUNT_ADJUSTED        DECIMAL [0] 0.25
1     PIN_FLD_AMOUNT_TAX_ADJUSTED    DECIMAL [0] 0
1     PIN_FLD_AMOUNT_TAXED           DECIMAL [0] 0
0 PIN_FLD_BAL_GRP_OBJ                   POID [0] 0.0.0.1 /balance_group 27496 3
0 PIN_FLD_ITEM_OBJ                      POID [0] 0.0.0.1 /item/adjustment 28568 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 20, used 3
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /event/session
220025470857535512 0
1     PIN_FLD_RESULT                    ENUM [0] 1
1     PIN_FLD_EVENTS                   ARRAY [0] allocated 1, used 1
2          PIN_FLD_POID                 POID [0] 0.0.0.1
/event/billing/adjustment/event
                                                 220025470857535576 0
0 PIN_FLD_ITEMS                        ARRAY [0] allocated 20, used 1
1     PIN_FLD_ITEM_OBJ                  POID [0] 0.0.0.1 /item/adjustment 28568 0
```

PCM_OP_AR_EVENT_DISPUTE

Opens a dispute against one or more events associated with an account. This opcode receives a batch of events on the input flist, and returns the disputed data on the output flist.

See the discussion on disputing events in *BRM Managing Accounts Receivable*.

Example 1–32 Sample Input Flist

```
0 PIN_FLD_EVENTS             ARRAY [0] allocated 20, used 1
1     PIN_FLD_POID            POID [0] 0.0.0.1 /event/session 219673627136631863 1
0 PIN_FLD_POID                POID [0] 0.0.0.1 /account 9627 0
0 PIN_FLD_END_T              TSTAMP [0] (1076998844) Mon Feb 16 22:20:44 2004
0 PIN_FLD_DISPUTES            ARRAY [0] allocated 20, used 2
1     PIN_FLD_PERCENT       DECIMAL [0] 10
1     PIN_FLD_RESOURCE_ID       INT [0] 840
0 PIN_FLD_PROGRAM_NAME         STR [0] "EventBrowser"
0 PIN_FLD_DESCR                STR [0] "Customer unaware of charges"
```

Example 1–33 Sample Output Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 9627 0
0 PIN_FLD_RESULT                        ENUM [0] 1
0 PIN_FLD_DESCR                          STR [0] "Successful"
0 PIN_FLD_DISPUTES                     ARRAY [0] allocated 5, used 5
1     PIN_FLD_RESOURCE_ID                INT [0] 840
1     PIN_FLD_AMOUNT                 DECIMAL [0] 2.5666667
1     PIN_FLD_AMOUNT_ADJUSTED        DECIMAL [0] -0.26
1     PIN_FLD_AMOUNT_TAX_ADJUSTED    DECIMAL [0] 0
1     PIN_FLD_AMOUNT_TAXED           DECIMAL [0] 0
0 PIN_FLD_BAL_GRP_OBJ                   POID [0] 0.0.0.1 /balance_group 10331 12
0 PIN_FLD_ITEM_OBJ                      POID [0] 0.0.0.1 /item/dispute 28168 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 3, used 3
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /event/session
219673627136631863 1
1     PIN_FLD_RESULT                    ENUM [0] 1
1     PIN_FLD_EVENTS                   ARRAY [0] allocated 1, used 1
2          PIN_FLD_POID                 POID [0] 0.0.0.1
/event/billing/dispute/event
                                                 220025470857535752 0
0 PIN_FLD_RESULTS                      ARRAY [1] allocated 4, used 4
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /event/billing/cycle/tax
                                                 219779180252904974 1
1     PIN_FLD_RESULT                    ENUM [0] 0
1     PIN_FLD_TYPE                      ENUM [0] 1
1     PIN_FLD_DESCR                      STR [0] "Nothing to adjust"
```

PCM_OP_AR_EVENT_SETTLEMENT

Settles dispute events associated with an account. This opcode receives dispute item POID in the input flists and retrieves the associated dispute events. It returns settlement data on the output flist.

See the discussion on settling disputed events in *BRM Managing Accounts Receivable*.

Example 1–34 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 14496 0
0 PIN_FLD_ITEM_OBJ           POID [0] 0.0.0.1 /item/dispute 14176 0
0 PIN_FLD_ADJUSTMENT_INFO   ARRAY [0] allocated 2, used 2
1     PIN_FLD_AMOUNT      DECIMAL [0] 0.5
1     PIN_FLD_RESOURCE_ID     INT [0] 840
0 PIN_FLD_PROGRAM_NAME        STR [0] "EventBrowser"
0 PIN_FLD_DESCR               STR [0] "Event settled in full"
```

Example 1–35 Sample Output Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 14496 0
0 PIN_FLD_RESULT                        ENUM [0] 1
0 PIN_FLD_DESCR                          STR [0] "Successful"
0 PIN_FLD_ADJUSTMENT_INFO              ARRAY [0] allocated 20, used 5
1     PIN_FLD_RESOURCE_ID               INT [0] 840
1     PIN_FLD_AMOUNT                DECIMAL [0] -1.5
1     PIN_FLD_AMOUNT_ADJUSTED       DECIMAL [0] -0.5
1     PIN_FLD_AMOUNT_TAX_ADJUSTED   DECIMAL [0] 0
1     PIN_FLD_AMOUNT_TAXED          DECIMAL [0] 0
0 PIN_FLD_BAL_GRP_OBJ                   POID [0] 0.0.0.1 /balance_group 15008 3
0 PIN_FLD_ITEM_OBJ                      POID [0] 0.0.0.1 /item/settlement 13536 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 20, used 3
1     PIN_FLD_POID                      POID [0] 0.0.0.1
/event/billing/dispute/event
                                                219269006857617248 0
1     PIN_FLD_RESULT                    ENUM [0] 1
1     PIN_FLD_EVENTS                   ARRAY [0] allocated 1, used 1
2          PIN_FLD_POID                 POID [0] 0.0.0.1
/event/billing/settlement/event
                                                219269006857616608 0
0 PIN_FLD_ITEMS                        ARRAY [0] allocated 20, used 1
1     PIN_FLD_ITEM_OBJ                  POID [0] 0.0.0.1 /item/settlement 13536 0
```

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; 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.

This opcode uses the PIN_FLD_FLAGS both as an input and output field. The values it uses are defined in the **pin_ar.h** file as follows:

```
#define PIN_AR_BILLED_ITEM 0x01
#define PIN_AR_UNBILLED_ITEM 0x02
```

The input value in PIN_FLD_FLAGS this opcode receives as input determines whether the opcode should mark the item as billed or unbilled. If this opcode does not receive PIN_FLD_FLAGS as an input, it marks the items as billed and unbilled.

It 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, the opcode selects both billed and unbilled items allocated to the bill, but does not mark the A/R items.

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

This opcode calls the **fm_ar_set_billed_unbilled_flag** which 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 contained PIN_FLD_FLAGS field and if a special item is allocated to the bill.

The special items that this opcode 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

See the discussion on retrieving A/R items that apply to a bill unit in *BRM Managing Accounts Receivable*.

Example 1–36 Sample Input Flist

This sample input flist calls the opcode for an account:

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_INCLUDE_CHILDREN               INT [0] 0
```

Example 1–37 Sample Output Flist

This output flist is the output from calling the opcode for an account:

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 15, used 15
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/adjustment 20232 2
1     PIN_FLD_ITEM_NO                    STR [0] "A1-2"
1     PIN_FLD_NAME                       STR [0] "Adjustment"
1     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -10
1     PIN_FLD_DUE                    DECIMAL [0] 0
1     PIN_FLD_TRANSFERED             DECIMAL [0] -10
1     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1098301549) Wed Oct 20 12:45:49
2004
1     PIN_FLD_CREATED_T               TSTAMP [0] (1098301549) Wed Oct 20 12:45:49
2004
1     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 16496 0
1     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 19568 0
1     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 19568 1
1     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.0  0 0
1     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
1     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-16496"
1     PIN_FLD_ALLOCATED              DECIMAL [0] -10
0 PIN_FLD_RESULTS                      ARRAY [1] allocated 15, used 15
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 18808 2
1     PIN_FLD_ITEM_NO                    STR [0] "D1-2"
1     PIN_FLD_NAME                       STR [0] "Dispute"
1     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -15
1     PIN_FLD_DUE                    DECIMAL [0] 0
1     PIN_FLD_TRANSFERED             DECIMAL [0] -15
1     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1098306824) Wed Oct 20 14:13:44
2004
1     PIN_FLD_CREATED_T               TSTAMP [0] (1098306824) Wed Oct 20 14:13:44
2004
1     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 16496 10
1     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 18032 0
1     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 18032 1
1     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 18160
1
1     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
1     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-16496"
1     PIN_FLD_ALLOCATED              DECIMAL [0] -15
```

Example 1–38 Sample Input Flist

This sample input flist calls the opcode for a single bill unit of an account:

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_AR_BILLINFO_OBJ               POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_BILLINFO_OBJ                  POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_INCLUDE_CHILDREN                NT [0] 0
```

Example 1–39 Sample Output Flist

This output flist is the output from calling the opcode for a single bill unit of an account:

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 16, used 16
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 19720 2
1     PIN_FLD_ITEM_NO                    STR [0] "D1-1"
1     PIN_FLD_NAME                       STR [0] "Dispute"
1     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -5
1     PIN_FLD_DUE                    DECIMAL [0] 0
1     PIN_FLD_TRANSFERED             DECIMAL [0] -5
1     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1097385738) Sat Oct  9 22:22:18
2004
1     PIN_FLD_CREATED_T               TSTAMP [0] (1097385738) Sat Oct  9 22:22:18
2004
1     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 16496 7
1     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 19568 0
1     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 19568 1
1     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 16624
1
1     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.1 /item/settlement 17032 1
1     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-16496"
1     PIN_FLD_ALLOCATED              DECIMAL [0] -5
1     PIN_FLD_POSTED_T                TSTAMP [0] (1098301603) Wed Oct 20 12:46:43
2004
```

PCM_OP_AR_GET_ACCT_BAL_SUMMARY

Retrieves the consolidated unapplied, open bill due, pending bill due, and total dispute balances for the all the bill units (**/billinfo** objects) in an account or for a specified **/billinfo** in an account.

See the discussion on retrieving a balance summary in *BRM Managing Accounts Receivable*.

Example 1–40 Sample Input Flist

The following sample calls the opcode for an account:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_INCLUDE_CHILDREN       INT [0] 0
```

Example 1–41 Sample Output Flist

The following sample is the output of calling the opcode for an account:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_BILLINFO_OBJ          POID [0] NULL poid pointer
0 PIN_FLD_AR_BILLINFO_OBJ       POID [0] NULL poid pointer
0 PIN_FLD_OPENBILL_DUE       DECIMAL [0] -17.05
0 PIN_FLD_UNAPPLIED_AMOUNT   DECIMAL [0] 0
0 PIN_FLD_TOTAL_RECORDS          INT [0] 0
0 PIN_FLD_PENDINGBILL_DUE    DECIMAL [0] 0
0 PIN_FLD_DISPUTED           DECIMAL [0] -15
0 PIN_FLD_BILL_OBJ              POID [0] NULL poid pointer
0 PIN_FLD_COUNT                  INT [0] 0
```

Example 1–42 Sample Input Flist

The following sample calls the opcode for a bill unit of an account:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_AR_BILLINFO_OBJ       POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_INCLUDE_CHILDREN       INT [0] 0
```

Example 1–43 Sample Output Flist

The following sample is the output of calling the opcode for a bill unit of an account:

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_AR_BILLINFO_OBJ       POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_BILL_OBJ              POID [0] 0.0.0.1 /bill 17008 0
0 PIN_FLD_COUNT                  INT [0] 0
0 PIN_FLD_OPENBILL_DUE       DECIMAL [0] -12
0 PIN_FLD_TOTAL_RECORDS          INT [0] 0
0 PIN_FLD_UNAPPLIED_AMOUNT   DECIMAL [0] 0
0 PIN_FLD_PENDINGBILL_DUE    DECIMAL [0] 0
0 PIN_FLD_DISPUTED           DECIMAL [0] 0
0 PIN_FLD_BILLINFO_OBJ          POID [0] 0.0.0.1 /billinfo 19568 0
```

PCM_OP_AR_GET_ACCT_BILLS

Retrieves the list of bills for all bill units (**/billinfo** objects) 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.

By default, this opcode 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 the PCM_OP_AR_GET_BILLS opcode 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, this opcode 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 On Demand"
  #define PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND "PIN Corrective Bill Now"
  ```

- 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.

See the discussion on retrieving a list of bills for a bill unit in *BRM Managing Accounts Receivable*.

Example 1–44 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_INCLUDE_CHILDREN         INT [0] 0
```

Example 1–45 Sample Output Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /account 16496 0
0 PIN_FLD_RESULTS                ARRAY [0] allocated 21, used 21
1     PIN_FLD_POID                POID [0] 0.0.0.1 /bill 16776 2
1     PIN_FLD_BILL_NO              STR [0] "B1-262"
1     PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 16496 0
1     PIN_FLD_BILLINFO_OBJ        POID [0] 0.0.0.1 /billinfo 19568 4
1     PIN_FLD_AR_BILLINFO_OBJ     POID [0] 0.0.0.1 /billinfo 19568 1
1     PIN_FLD_PARENT              POID [0] 0.0.0.0  0 0
1     PIN_FLD_ADJUSTED         DECIMAL [0] -15
1     PIN_FLD_DISPUTED         DECIMAL [0] 0
1     PIN_FLD_RECVD            DECIMAL [0] 0
1     PIN_FLD_TRANSFERED       DECIMAL [0] 0
1     PIN_FLD_WRITEOFF         DECIMAL [0] 0
1     PIN_FLD_DUE              DECIMAL [0] -12
1     PIN_FLD_CURRENT_TOTAL    DECIMAL [0] -12
1     PIN_FLD_SUBORDS_TOTAL    DECIMAL [0] 0
1     PIN_FLD_PREVIOUS_TOTAL   DECIMAL [0] 0
1     PIN_FLD_CREATED_T         TSTAMP [0] (1098301630) Wed Oct 20 12:47:10 2004
1     PIN_FLD_START_T           TSTAMP [0] (1098301630) Wed Oct 20 12:47:10 2004
1     PIN_FLD_END_T             TSTAMP [0] (1098301630) Wed Oct 20 12:47:10 2004
1     PIN_FLD_DUE_T             TSTAMP [0] (1100897230) Fri Nov 19 12:47:10 2004
1     PIN_FLD_TOTALS           DECIMAL [0] 3
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (1)"
0 PIN_FLD_RESULTS                ARRAY [1] allocated 21, used 21
1     PIN_FLD_POID                POID [0] 0.0.0.1 /bill 19320 2
1     PIN_FLD_BILL_NO              STR [0] "B1-276"
1     PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 16496 0
1     PIN_FLD_BILLINFO_OBJ        POID [0] 0.0.0.1 /billinfo 18032 3
1     PIN_FLD_AR_BILLINFO_OBJ     POID [0] 0.0.0.1 /billinfo 18032 1
1     PIN_FLD_PARENT              POID [0] 0.0.0.0  0 0
1     PIN_FLD_ADJUSTED         DECIMAL [0] 0
1     PIN_FLD_DISPUTED         DECIMAL [0] -15
1     PIN_FLD_RECVD            DECIMAL [0] 0
1     PIN_FLD_TRANSFERED       DECIMAL [0] 0
1     PIN_FLD_WRITEOFF         DECIMAL [0] 0
1     PIN_FLD_DUE              DECIMAL [0] -5.05
1     PIN_FLD_CURRENT_TOTAL    DECIMAL [0] -5.05
1     PIN_FLD_SUBORDS_TOTAL    DECIMAL [0] 0
1     PIN_FLD_PREVIOUS_TOTAL   DECIMAL [0] 0
1     PIN_FLD_CREATED_T         TSTAMP [0] (1098307659) Wed Oct 20 14:27:39 2004
1     PIN_FLD_START_T           TSTAMP [0] (1098307659) Wed Oct 20 14:27:39 2004
1     PIN_FLD_END_T             TSTAMP [0] (1098307659) Wed Oct 20 14:27:39 2004
1     PIN_FLD_DUE_T             TSTAMP [0] (1100903259) Fri Nov 19 14:27:39 2004
1     PIN_FLD_TOTALS           DECIMAL [0] 9.95
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (2)"
```

PCM_OP_AR_GET_ACTION_ITEMS

Retrieves the list of A/R items applied to a bill unit (**/billinfo** object).

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.

See the discussion on retrieving A/R items that apply to a bill unit in*BRM Managing Accounts Receivable*.

Example 1–46 Sample Input Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /account 0 0
0 PIN_FLD_AR_BILLINFO            POID [0] 0.0.0.1 /account 0 0
0 PIN_FLD_THRESHOLD               INT [0] 20
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /account 12298 18
0 PIN_FLD_AR_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 12298 18
0 PIN_FLD_INCLUDE_CHILDREN        INT [0] 1
0 PIN_FLD_POID_TYPE               STR [0] "'/item/settlement','/item/dispute',
                                          '/item/writeoff','/item/refund','/item/a
djustment'"
```

Note: PIN_FLD_POID is required because every flist must have a POID (it can be any POID).

Example 1–47 Sample Output Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 0 0
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 20, used 13
1     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/adjustment 19950 1
1     PIN_FLD_ITEM_NO                    STR [0] "A-11"
1     PIN_FLD_NAME                       STR [0] "Adjustment"
1     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -10
1     PIN_FLD_DUE                    DECIMAL [0] -10
1     PIN_FLD_TRANSFERED             DECIMAL [0] 0
1     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1005844209) Thu Nov 15 09:10:09
2001
1     PIN_FLD_CREATED_T               TSTAMP [0] (1003165567) Mon Oct 15 10:06:07
2001
1     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 12298 16
1     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.0  0 0
1     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
1     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-12298"
1     PIN_FLD_ALLOCATED              DECIMAL [0] 0
```

PCM_OP_AR_GET_BAL_SUMMARY

Retrieves the unapplied, open bill due, pending bill due, and total dispute balances for a given bill unit (**/billinfo** object).

See the discussion on retrieving a balance summary in *BRM Managing Accounts Receivable*.

Example 1–48 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /billinfo 48991 3
0 PIN_FLD_AR_BILLINFO_OBJ      POID [0] 0.0.0.1 /billinfo 48991 2
0 PIN_FLD_INCLUDE_CHILDREN      INT [0] 1
```

Example 1–49 Sample Output Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /billinfo 48991 3
0 PIN_FLD_BILLINFO_OBJ        POID [0] 0.0.0.1 /billinfo 48991 3
0 PIN_FLD_AR_BILLINFO_OBJ     POID [0] 0.0.0.1 /billinfo 48991 2
0 PIN_FLD_BILL_OBJ            POID [0] 0.0.0.1 /bill 55148 0
0 PIN_FLD_COUNT               INT [0] 0
0 PIN_FLD_OPENBILL_DUE        DECIMAL [0] 42.9
0 PIN_FLD_TOTAL_RECORDS       INT [0] 0
0 PIN_FLD_UNAPPLIED_AMOUNT    DECIMAL [0] 0
0 PIN_FLD_PENDINGBILL_DUE     DECIMAL [0] 0
0 PIN_FLD_DISPUTED            DECIMAL [0] 0
0 PIN_FLD_DISPUTE_TYPE        ENUM [0] 0
0 PIN_FLD_ITEM_PENDING_FLAGS   ENUM [0] 0
```

PCM_OP_AR_GET_BILL_ITEMS

Retrieves the list of bill items for a bill unit (**/billinfo** object).

> **Note:** Bill items are referred to as *item charges* in Customer Center.

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.

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.

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, this opcode 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, this opcode 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, the 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, this opcode 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, the 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, the opcode 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, the opcode does not return the summation of the billed or unbilled amounts.

See the discussion on retrieving a list of bill items for a bill unit in *BRM Managing Accounts Receivable*.

Example 1–50 Sample Input Flist

```
0 PIN_FLD_THRESHOLD           INT [0] 20
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 12650 16
0 PIN_FLD_AR_BILLINFO_OBJ    POID [0] 0.0.0.1 /billinfo 12554 1
0 PIN_FLD_INCLUDE_CHILDREN    INT [0] 1
0 PIN_FLD_BILL_OBJ           POID [0] 0.0.0.1 /bill 13162 0
```

Example 1–51 Sample Output Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_RESULTS            ARRAY [0] allocated 27, used 14
1     PIN_FLD_POID             POID [0] 0.0.0.1 /item/misc 14698 1
1     PIN_FLD_ITEM_NO           STR [0] ""
1     PIN_FLD_NAME              STR [0] "Usage"
1     PIN_FLD_ITEM_TOTAL    DECIMAL [0] 0
1     PIN_FLD_DUE           DECIMAL [0] 0
1     PIN_FLD_TRANSFERED    DECIMAL [0] 0
1     PIN_FLD_ADJUSTED      DECIMAL [0] 0
1     PIN_FLD_DISPUTED      DECIMAL [0] 0
1     PIN_FLD_RECVD         DECIMAL [0] 0
1     PIN_FLD_WRITEOFF      DECIMAL [0] 0
1     PIN_FLD_EFFECTIVE_T    TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 12650 0
1     PIN_FLD_BILLINFO_OBJ     POID [0] 0.0.0.0 /billinfo 12554 1
1     PIN_FLD_ACCOUNT_NO        STR [0] "0.0.0.1-12650"
0 PIN_FLD_RESULTS             ARRAY [1] allocated 27, used 14
1     PIN_FLD_POID             POID [0] 0.0.0.1 /item/cycle_forward 14186 1
1     PIN_FLD_ITEM_NO           STR [0] ""
1     PIN_FLD_NAME              STR [0] "Cycle forward"
1     PIN_FLD_ITEM_TOTAL    DECIMAL [0] 100
1     PIN_FLD_DUE           DECIMAL [0] 100
1     PIN_FLD_TRANSFERED    DECIMAL [0] 0
1     PIN_FLD_ADJUSTED      DECIMAL [0] 0
1     PIN_FLD_DISPUTED      DECIMAL [0] 0
1     PIN_FLD_RECVD         DECIMAL [0] 0
1     PIN_FLD_WRITEOFF      DECIMAL [0] 0
1     PIN_FLD_EFFECTIVE_T    TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 12650 10
1     PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.1 /service/ip 15210 0
1     PIN_FLD_ACCOUNT_NO        STR [0] "0.0.0.1-12650"
```

PCM_OP_AR_GET_BILLS

Retrieves a list of bills for a bill unit (**/billinfo** object) based on the start time and end time search criteria.

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.

For corrective bills, this opcode 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.
- This opcode 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 On Demand"
    #define PIN_OBJ_NAME_CORRECTIVE_BILL_ON_DEMAND "PIN Corrective Bill Now"
    ```

  - 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.

See the discussion on retrieving a list of bills for a bill unit in *BRM Managing Accounts Receivable*.

Example 1–52 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /bill 12810 0
0 PIN_FLD_AR_BILLINFO_OBJ    POID [0] 0.0.0.1 /billinfo 12298 18
0 PIN_FLD_INCLUDE_CHILDREN    INT [0] 1
```

Example 1–53 Sample Output Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /bill 12810 0
0 PIN_FLD_RESULTS               ARRAY [0] allocated 31, used 18
1     PIN_FLD_POID                POID [0] 0.0.0.1 /bill 12810 1
1     PIN_FLD_BILL_NO              STR [0] "B1-22"
1     PIN_FLD_BILLINFO_OBJ        POID [0] 0.0.0.1 /billinfo 12298 0
1     PIN_FLD_PARENT              POID [0] 0.0.0.0 0 0
```

```
1     PIN_FLD_ADJUSTED         DECIMAL [0] 0
1     PIN_FLD_DISPUTED         DECIMAL [0] 0
1     PIN_FLD_RECVD            DECIMAL [0] 0
1     PIN_FLD_TRANSFERED       DECIMAL [0] 0
1     PIN_FLD_WRITEOFF         DECIMAL [0] 0
1     PIN_FLD_DUE              DECIMAL [0] 0
1     PIN_FLD_CURRENT_TOTAL    DECIMAL [0] NULL
1     PIN_FLD_SUBORDS_TOTAL    DECIMAL [0] NULL
1     PIN_FLD_PREVIOUS_TOTAL   DECIMAL [0] NULL
1     PIN_FLD_CREATED_T         TSTAMP [0] (1002738418) Wed Oct 10 11:26:58 2001
1     PIN_FLD_END_T             TSTAMP [0] (0) <null>
1     PIN_FLD_AR_BILLINFO_OBJ     POID [0] 0.0.0.1 /billinfo 12298 18
1     PIN_FLD_DUE_T             TSTAMP [0] (0) <null>
1     PIN_FLD_TOTALS           DECIMAL [0] 0
```

PCM_OP_AR_GET_DISPUTE_DETAILS

Retrieves all event-level and item-level disputes and the aggregated amount of each resource for the dispute events associated with a dispute item (**/item/dispute** object). This opcode is called by Customer Center.

See the discussion on retrieving dispute details for a bill unit in *BRM Managing Accounts Receivable*.

Example 1–54 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 13953 13
0 PIN_FLD_AR_BILLINFO_OBJ               POID [0] 0.0.0.1 /billinfo 14721 1
0 PIN_FLD_INCLUDE_CHILDREN               INT [0] 0
0 PIN_FLD_STATUS                        ENUM [0] 2
```

Example 1–55 Sample Output Flist

The following sample includes comments to help you interpret the flist. These comments are prefaced by an exclamation mark (!) and do not normally appear in the flist.

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /billinfo 14721 1
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 29, used 29
1   PIN_FLD_POID                        POID [0] 0.0.0.1 /item/cycle_forward 14401
10
1   PIN_FLD_ITEM_NO                      STR [0] "B1-33,1"
1   PIN_FLD_NAME                         STR [0] "Cycle forward"
1   PIN_FLD_CREATED_T                 TSTAMP [0] (1076127297) Fri Feb  6 20:14:57
2004
1   PIN_FLD_MOD_T                     TSTAMP [0] (1076390230) Mon Feb  9 21:17:10
2004
1   PIN_FLD_EFFECTIVE_T               TSTAMP [0] (1088413545) Mon Jun 28 02:05:45
2004
1   PIN_FLD_CLOSED_T                  TSTAMP [0] (0) <null>
1   PIN_FLD_DUE_T                     TSTAMP [0] (1091005545) Wed Jul 28 02:05:45
2004
1   PIN_FLD_STATUS                       ENUM [0] 2
1   PIN_FLD_ACCOUNT_OBJ                  POID [0] 0.0.0.1 /account 13953 7
1   PIN_FLD_BILLINFO_OBJ                 POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_AR_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_BILL_OBJ                     POID [0] 0.0.0.1 /bill 15681 0
1   PIN_FLD_AR_BILL_OBJ                  POID [0] 0.0.0.1 /bill 15681 1
1   PIN_FLD_SERVICE_OBJ                  POID [0] 0.0.0.1 /service/ip 16257 0
1   PIN_FLD_ITEM_TOTAL               DECIMAL [0] 30
1   PIN_FLD_ADJUSTED                 DECIMAL [0] -2
1   PIN_FLD_DISPUTED                 DECIMAL [0] -1.8
1   PIN_FLD_DUE                      DECIMAL [0] 26.2
1   PIN_FLD_RECVD                    DECIMAL [0] 0
1   PIN_FLD_TRANSFERED               DECIMAL [0] 0
1   PIN_FLD_WRITEOFF                 DECIMAL [0] 0
1   PIN_FLD_CURRENCY                     INT [0] 840
1   PIN_FLD_CURRENCY_SECONDARY           INT [0] 0

! Transfer arrays for the first event-level disputes

1   PIN_FLD_TRANSFERS_OUT              ARRAY [0] allocated 21, used 21
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 16273 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-11"
2     PIN_FLD_NAME                       STR [0] "Dispute"
```

```
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088592223) Wed Jun 30 03:43:43
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076305995) Sun Feb  8 21:53:15
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
7
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] ""

! Arrays for the first event-level dispute

2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 12369 0
3       PIN_FLD_DESCR                    STR [0] ""

! Array of aggregated resource amounts for the first event-level dispute

2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [0] allocated 2, used 2
3       PIN_FLD_RESOURCE_ID              INT [0] 840
3       PIN_FLD_AMOUNT                DECIMAL [0] -1
2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [1] allocated 2, used 2
3       PIN_FLD_RESOURCE_ID              INT [0] 1000020
3       PIN_FLD_AMOUNT                DECIMAL [0] 0

! Transfer arrays for the second event-level disputes

1   PIN_FLD_TRANSFERS_OUT              ARRAY [1] allocated 21, used 21
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 13249 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-8"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419110) Mon Jun 28 03:38:30
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132881) Fri Feb  6 21:48:01
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
0
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] ""
```

```
! Arrays for the second event-level dispute

2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 15297 0
3       PIN_FLD_DESCR                     STR [0] ""

! Array of aggregated resource amounts for the second event-level dispute

2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [0] allocated 2, used 2
3       PIN_FLD_RESOURCE_ID              INT [0] 840
3       PIN_FLD_AMOUNT                DECIMAL [0] -1
2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [1] allocated 2, used 2
3       PIN_FLD_RESOURCE_ID              INT [0] 1000020
3       PIN_FLD_AMOUNT                DECIMAL [0] -1

! Transfer array for item-level dispute

1   PIN_FLD_TRANSFERS_OUT              ARRAY [2] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 12737 2
2     PIN_FLD_ITEM_NO                    STR [0] "D1-26"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] 0.1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] 0.1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419006) Mon Jun 28 03:36:46
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132777) Fri Feb  6 21:46:17
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 7
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] 0.1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
5
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 0
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_EVENT_OBJ                 POID [0] 0.0.0.1
/event/billing/dispute/item 14785 0
2     PIN_FLD_DESCR                      STR [0] ""
```

PCM_OP_AR_GET_DISPUTES

Retrieves details of all disputed bill items for a bill unit (**/billinfo** object).

> **Note:** Bill items are referred to as *item charges* in Customer Center. Customer Center uses this opcode to get the list of disputes for a bill unit.

See the discussion on retrieving dispute details for a bill unit in *BRM Managing Accounts Receivable*.

Example 1–56 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 13953 13
0 PIN_FLD_AR_BILLINFO_OBJ               POID [0] 0.0.0.1 /billinfo 14721 1
0 PIN_FLD_INCLUDE_CHILDREN               INT [0] 0
0 PIN_FLD_STATUS                        ENUM [0] 2
```

Example 1–57 Sample Output Flist

The following sample includes comments to help you interpret the flist. These comments are prefaced by an exclamation mark (!) and do not normally appear in the flist.

```
0 PIN_FLD_POID                           POID [0] 0.0.0.1 /billinfo 14721 1
0 PIN_FLD_RESULTS                       ARRAY [0] allocated 29, used 29
1   PIN_FLD_POID                         POID [0] 0.0.0.1 /item/cycle_forward 14401
10
1   PIN_FLD_ITEM_NO                       STR [0] "B1-33,1"
1   PIN_FLD_NAME                          STR [0] "Cycle forward"
1   PIN_FLD_CREATED_T                  TSTAMP [0] (1076127297) Fri Feb  6 20:14:57
2004
1   PIN_FLD_MOD_T                      TSTAMP [0] (1076390230) Mon Feb  9 21:17:10
2004
1   PIN_FLD_EFFECTIVE_T                TSTAMP [0] (1088413545) Mon Jun 28 02:05:45
2004
1   PIN_FLD_CLOSED_T                   TSTAMP [0] (0) <null>
1   PIN_FLD_DUE_T                      TSTAMP [0] (1091005545) Wed Jul 28 02:05:45
2004
1   PIN_FLD_STATUS                       ENUM [0] 2
1   PIN_FLD_ACCOUNT_OBJ                  POID [0] 0.0.0.1 /account 13953 7
1   PIN_FLD_BILLINFO_OBJ                 POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_AR_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_BILL_OBJ                     POID [0] 0.0.0.1 /bill 15681 0
1   PIN_FLD_AR_BILL_OBJ                  POID [0] 0.0.0.1 /bill 15681 1
1   PIN_FLD_SERVICE_OBJ                  POID [0] 0.0.0.1 /service/ip 16257 0
1   PIN_FLD_ITEM_TOTAL                DECIMAL [0] 30
1   PIN_FLD_ADJUSTED                  DECIMAL [0] -2
1   PIN_FLD_DISPUTED                  DECIMAL [0] -1.8
1   PIN_FLD_DUE                       DECIMAL [0] 26.2
1   PIN_FLD_RECVD                     DECIMAL [0] 0
1   PIN_FLD_TRANSFERED                DECIMAL [0] 0
1   PIN_FLD_WRITEOFF                  DECIMAL [0] 0
1   PIN_FLD_CURRENCY                      INT [0] 840
1   PIN_FLD_CURRENCY_SECONDARY            INT [0] 0

! Transfer arrays for event-level disputes
```

```
1   PIN_FLD_TRANSFERS_OUT              ARRAY [0] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 16273 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-11"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088592223) Wed Jun 30 03:43:43
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076305995) Sun Feb  8 21:53:15
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
7
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] ""
2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 12369 0
3       PIN_FLD_DESCR                    STR [0] ""
1   PIN_FLD_TRANSFERS_OUT              ARRAY [1] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 13249 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-8"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419110) Mon Jun 28 03:38:30
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132881) Fri Feb  6 21:48:01
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
0
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] ""
2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 15297 0
3       PIN_FLD_DESCR                    STR [0] ""

! Transfer array for item-level dispute

1   PIN_FLD_TRANSFERS_OUT              ARRAY [2] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 12737 2
2     PIN_FLD_ITEM_NO                    STR [0] "D1-26"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] 0.1
```

```
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] 0.1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419006) Mon Jun 28 03:36:46
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132777) Fri Feb  6 21:46:17
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 7
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] 0.1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
5
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 0
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_EVENT_OBJ                 POID [0] 0.0.0.1
/event/billing/dispute/item 14785 0
2     PIN_FLD_DESCR                      STR [0] ""
```

PCM_OP_AR_GET_ITEM_DETAIL

Retrieves details for the specified A/R item or bill item. The data retrieved does not include noncurrency resources.

For example, Customer Center uses this opcode to retrieve detailed information about a specific write-off or usage item.

See the discussion on retrieving details about a specific A/R item or bill item in *BRM Managing Accounts Receivable*.

Example 1–58 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /item/cycle_forward 14401
10
```

Example 1–59 Sample Output Flist

The following sample includes comments to help you interpret the flist. These comments are prefaced by an exclamation mark (!) and do not normally appear in the flist.

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /item/cycle_forward 14401
10
0 PIN_FLD_RESULTS                      ARRAY [0] allocated 30, used 30
1   PIN_FLD_POID                        POID [0] 0.0.0.1 /item/cycle_forward 14401
10
1   PIN_FLD_ITEM_NO                      STR [0] "B1-33,1"
1   PIN_FLD_NAME                         STR [0] "Cycle forward"
1   PIN_FLD_CREATED_T                 TSTAMP [0] (1076127297) Fri Feb  6 20:14:57
2004
1   PIN_FLD_MOD_T                     TSTAMP [0] (1076390230) Mon Feb  9 21:17:10
2004
1   PIN_FLD_EFFECTIVE_T               TSTAMP [0] (1088413545) Mon Jun 28 02:05:45
2004
1   PIN_FLD_CLOSED_T                  TSTAMP [0] (0) <null>
1   PIN_FLD_DUE_T                     TSTAMP [0] (1091005545) Wed Jul 28 02:05:45
2004
1   PIN_FLD_STATUS                       ENUM [0] 2
1   PIN_FLD_ACCOUNT_OBJ                  POID [0] 0.0.0.1 /account 13953 7
1   PIN_FLD_BILLINFO_OBJ                 POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_AR_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
1   PIN_FLD_BILL_OBJ                     POID [0] 0.0.0.1 /bill 15681 0
1   PIN_FLD_AR_BILL_OBJ                  POID [0] 0.0.0.1 /bill 15681 1
1   PIN_FLD_SERVICE_OBJ                  POID [0] 0.0.0.1 /service/ip 16257 0
1   PIN_FLD_ITEM_TOTAL                DECIMAL [0] 30
1   PIN_FLD_ADJUSTED                  DECIMAL [0] -2
1   PIN_FLD_DISPUTED                  DECIMAL [0] -1.8
1   PIN_FLD_DUE                       DECIMAL [0] 26.2
1   PIN_FLD_RECVD                     DECIMAL [0] 0
1   PIN_FLD_TRANSFERED                DECIMAL [0] 0
1   PIN_FLD_WRITEOFF                  DECIMAL [0] 0
1   PIN_FLD_CURRENCY                      INT [0] 840
1   PIN_FLD_CURRENCY_SECONDARY            INT [0] 0
1   PIN_FLD_ACCOUNT_NO                    STR [0] "0.0.0.1-13953"

! Transfer arrays for event-level disputes

1   PIN_FLD_TRANSFERS_INTO             ARRAY [0] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 16273 2
```

```
2     PIN_FLD_ITEM_NO                    STR [0] "A1-11"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088592223) Wed Jun 30 03:43:43
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076305995) Sun Feb  8 21:53:15
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
7
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_RESOURCE_IMPACTED         ENUM [0] 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] "[Customer not satisfied with
service]"
2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 12369 0
3       PIN_FLD_DESCR                    STR [0] "[Customer not satisfied with
service]"
1   PIN_FLD_TRANSFERS_INTO             ARRAY [1] allocated 19, used 19
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 13249 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-8"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] -1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419110) Mon Jun 28 03:38:30
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132881) Fri Feb  6 21:48:01
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 11
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] -1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
0
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_RESOURCE_IMPACTED         ENUM [0] 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_DESCR                      STR [0] "[Customer not satisfied with
service]"
2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event 15297 0
3       PIN_FLD_DESCR                    STR [0] "[Customer not satisfied with
service]"

! Transfer array for item-level dispute

1   PIN_FLD_TRANSFERS_INTO             ARRAY [2] allocated 19, used 19
```

```
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 12737 2
2     PIN_FLD_ITEM_NO                    STR [0] "D1-26"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] 0.1
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] 0.1
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1088419006) Mon Jun 28 03:36:46
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1076132777) Fri Feb  6 21:46:17
2004
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 13953 7
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 14721 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 14721 2
2     PIN_FLD_ALLOCATED              DECIMAL [0] 0.1
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 14401
5
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_RESOURCE_IMPACTED         ENUM [0] 0
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-13953"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 0
2     PIN_FLD_RESOURCE_ID                INT [0] 840
2     PIN_FLD_EVENT_OBJ                 POID [0] 0.0.0.1
/event/billing/dispute/item 14785 0
2     PIN_FLD_DESCR                      STR [0] "[Customer not satisfied with
service]"
```

PCM_OP_AR_GET_ITEMS

Retrieves details for the specified A/R item or bill item. The data retrieved includes both currency and noncurrency resources. It also includes the aggregated amount of each resource for the events in the A/R or bill item.

For example, Customer Center uses this opcode to retrieve detailed information on adjustments, disputes, and settlements.

See the discussion on retrieving details about a specific A/R item or bill item in *BRM Managing Accounts Receivable*.

Example 1–60 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /item/cycle_forward 8371
1
```

Example 1–61 Sample Output Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /item/cycle_forward 8371
1
0 PIN_FLD_RESULTS                        ARRAY [0] allocated 28, used 28
1   PIN_FLD_POID                          POID [0] 0.0.0.1 /item/cycle_forward 8371
3
1   PIN_FLD_ITEM_NO                        STR [0] "B1-41,2"
1   PIN_FLD_NAME                           STR [0] "Cycle forward"
1   PIN_FLD_CREATED_T                   TSTAMP [0] (1109881823) Thu Mar  3 12:30:23
2005
1   PIN_FLD_MOD_T                       TSTAMP [0] (1109893810) Thu Mar  3 15:50:10
2005
1   PIN_FLD_EFFECTIVE_T                 TSTAMP [0] (1106726400) Wed Jan 26 00:00:00
2005
1   PIN_FLD_CLOSED_T                    TSTAMP [0] (0) <null>
1   PIN_FLD_DUE_T                       TSTAMP [0] (1109318400) Fri Feb 25 00:00:00
2005
1   PIN_FLD_STATUS                        ENUM [0] 2
1   PIN_FLD_ACCOUNT_OBJ                   POID [0] 0.0.0.1 /account 10787 14
1   PIN_FLD_BILLINFO_OBJ                  POID [0] 0.0.0.1 /billinfo 10531 0
1   PIN_FLD_AR_BILLINFO_OBJ               POID [0] 0.0.0.1 /billinfo 10531 0
1   PIN_FLD_BILL_OBJ                      POID [0] 0.0.0.1 /bill 9171 0
1   PIN_FLD_AR_BILL_OBJ                   POID [0] 0.0.0.1 /bill 9171 0
1   PIN_FLD_SERVICE_OBJ                   POID [0] 0.0.0.1 /service/ip 9379 0
1   PIN_FLD_ITEM_TOTAL                 DECIMAL [0] 100
1   PIN_FLD_ADJUSTED                   DECIMAL [0] 0
1   PIN_FLD_DISPUTED                   DECIMAL [0] -2
1   PIN_FLD_DUE                        DECIMAL [0] 98
1   PIN_FLD_RECVD                      DECIMAL [0] 0
1   PIN_FLD_TRANSFERED                 DECIMAL [0] 0
1   PIN_FLD_WRITEOFF                   DECIMAL [0] 0
1   PIN_FLD_CURRENCY                       INT [0] 840
1   PIN_FLD_CURRENCY_SECONDARY             INT [0] 0
1   PIN_FLD_ACCOUNT_NO                     STR [0] "0.0.0.1-10787"

! Transfer array for an event-level currency dispute

1   PIN_FLD_TRANSFERS_INTO              ARRAY [0] allocated 22, used 22
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/dispute 10267 2
2     PIN_FLD_ITEM_NO                    STR [0] "A1-8"
2     PIN_FLD_NAME                       STR [0] "Dispute"
2     PIN_FLD_ITEM_TOTAL              DECIMAL [0] -2
```

```
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] -2
2     PIN_FLD_EFFECTIVE_T             TSTAMP [0] (1098855103) Tue Oct 26 22:31:43
2004
2     PIN_FLD_CREATED_T               TSTAMP [0] (1109893808) Thu Mar  3 15:50:08
2005
2     PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 10787 18
2     PIN_FLD_BILLINFO_OBJ              POID [0] 0.0.0.1 /billinfo 10531 0
2     PIN_FLD_AR_BILLINFO_OBJ           POID [0] 0.0.0.1 /billinfo 10531 1
2     PIN_FLD_ALLOCATED              DECIMAL [0] -2
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 8371
2
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0
2     PIN_FLD_RESOURCE_IMPACTED         ENUM [0] 0
2     PIN_FLD_RESOURCE_ID                INT [0] 840

! Array for the event-level currency dispute

2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/dispute/event
                                                225971629740532763 0
3       PIN_FLD_DESCR                    STR [0] "[Customer not satisfied with
service]"
2     PIN_FLD_ACCOUNT_NO                 STR [0] "0.0.0.1-10787"
2     PIN_FLD_DISPUTE_TYPE              ENUM [0] 1
2     PIN_FLD_DESCR                      STR [0] "[Customer not satisfied with
service]"

! Array of aggregated resources for the event-level currency dispute

2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [0] allocated 4, used 4
3       PIN_FLD_RESOURCE_ID              INT [0] 840
3       PIN_FLD_AMOUNT                DECIMAL [0] 100
3       PIN_FLD_DISCOUNT              DECIMAL [0] 0
3       PIN_FLD_DISPUTED              DECIMAL [0] -2
2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [1] allocated 3, used 3
3       PIN_FLD_RESOURCE_ID              INT [0] 1000020
3       PIN_FLD_AMOUNT                DECIMAL [0] -200
3       PIN_FLD_DISCOUNT              DECIMAL [0] 0
1   PIN_FLD_TAX                       DECIMAL [0] 0

! Transfer array for an event-level noncurrency adjustment

1   PIN_FLD_TRANSFERS_INTO             ARRAY [1] allocated 37, used 37
2     PIN_FLD_POID                      POID [0] 0.0.0.1 /item/adjustment 10779 0
2     PIN_FLD_ITEM_NO                    STR [0] "A1-9"
2     PIN_FLD_NAME                       STR [0] "Adjustment"
2     PIN_FLD_CREATED_T                TSTAMP [0] (1109893932) Thu Mar  3 15:52:12
2005
2     PIN_FLD_MOD_T                    TSTAMP [0] (1109893932) Thu Mar  3 15:52:12
2005
2     PIN_FLD_EFFECTIVE_T              TSTAMP [0] (1098855227) Tue Oct 26 22:33:47
2004
2     PIN_FLD_CLOSED_T                 TSTAMP [0] (0) <null>
2     PIN_FLD_DUE_T                    TSTAMP [0] (0) <null>
2     PIN_FLD_STATUS                     ENUM [0] 2
2     PIN_FLD_ACCOUNT_OBJ                POID [0] 0.0.0.1 /account 10787 19
2     PIN_FLD_BILLINFO_OBJ               POID [0] 0.0.0.1 /billinfo 10531 0
2     PIN_FLD_AR_BILLINFO_OBJ            POID [0] 0.0.0.1 /billinfo 10531 1
```

```
2     PIN_FLD_BILL_OBJ                  POID [0] 0.0.0.0  0 0
2     PIN_FLD_AR_BILL_OBJ               POID [0] 0.0.0.0  0 0
2     PIN_FLD_SERVICE_OBJ               POID [0] 0.0.0.0  0 0
2     PIN_FLD_ITEM_TOTAL             DECIMAL [0] 0
2     PIN_FLD_ADJUSTED               DECIMAL [0] 0
2     PIN_FLD_DISPUTED               DECIMAL [0] 0
2     PIN_FLD_DUE                    DECIMAL [0] 0
2     PIN_FLD_RECVD                  DECIMAL [0] 0
2     PIN_FLD_TRANSFERED             DECIMAL [0] 0
2     PIN_FLD_WRITEOFF               DECIMAL [0] 0
2     PIN_FLD_CURRENCY                   INT [0] 840
2     PIN_FLD_CURRENCY_SECONDARY         INT [0] 0
2     PIN_FLD_RESOURCE_ID                INT [0] 1000020
2     PIN_FLD_RELATED_BILL_ITEM_OBJ     POID [0] 0.0.0.1 /item/cycle_forward 8371
0
2     PIN_FLD_RESOURCE_IMPACTED         ENUM [0] 0
2     PIN_FLD_ADJUSTMENT_TYPE           ENUM [0] 1
2     PIN_FLD_DESCR                      STR [0] "[Customer not satisfied with
service] "
2     PIN_FLD_RELATED_ACTION_ITEM_OBJ   POID [0] 0.0.0.0  0 0

! Array for the event-level noncurrency adjustment

2     PIN_FLD_EVENTS                   ARRAY [0] allocated 2, used 2
3       PIN_FLD_POID                    POID [0] 0.0.0.1
/event/billing/adjustment/event
                                                225971629740533275 0
3       PIN_FLD_DESCR                    STR [0] "[Customer not satisfied with
service]"
2     PIN_FLD_EVENT_OBJ                 POID [0] 0.0.0.1
/event/billing/adjustment/event
                                                225971629740533275 0
2     PIN_FLD_ACCOUNT_NO                STR [0] "0.0.0.1-10787"
2     PIN_FLD_TAX                    DECIMAL [0] 0

! Array of aggregated resources for the event-level noncurrency adjustment

2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [0] allocated 3, used 3
3       PIN_FLD_RESOURCE_ID               INT [0] 840
3       PIN_FLD_AMOUNT                DECIMAL [0] 100
3       PIN_FLD_DISCOUNT              DECIMAL [0] 0
2     PIN_FLD_AGGREGATE_AMOUNTS        ARRAY [1] allocated 4, used 4
3       PIN_FLD_RESOURCE_ID               INT [0] 1000020
3       PIN_FLD_AMOUNT                DECIMAL [0] -200
3       PIN_FLD_DISCOUNT              DECIMAL [0] 0
3       PIN_FLD_ADJUSTED              DECIMAL [0] -4
2     PIN_FLD_ALLOCATED               DECIMAL [0] 0
```

PCM_OP_AR_ITEM_ADJUSTMENT

Makes adjustments against items in an A/R bill.

See the discussion on adjusting items in *BRM Managing Accounts Receivable*.

Example 1–62 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/misc 27448 5
0 PIN_FLD_AMOUNT        DECIMAL [0] -2
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DESCR             STR [0] "Bad quality of service"
0 PIN_FLD_STR_VERSION       INT [0] 15
0 PIN_FLD_STRING_ID         INT [0] 3
0 PIN_FLD_FLAGS             INT [0] 2
```

Example 1–63 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/misc 27448 5
0 PIN_FLD_RESULT           ENUM [0] 3
0 PIN_FLD_DESCR             STR [0] "Successful"
0 PIN_FLD_RESULTS         ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/adjustment/item
                                     220025470857536632 0
0 PIN_FLD_RESULTS         ARRAY [1] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/item/transfer
                                     220025470857536120 0
```

PCM_OP_AR_ITEM_DISPUTE

Files a dispute against an item on a bill.

See the discussion on disputing items in *BRM Managing Accounts Receivable*.

Example 1–64 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/cycle_forward 14672 2
0 PIN_FLD_AMOUNT        DECIMAL [0] -0.5
0 PIN_FLD_CURRENCY          INT [0] 840
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DESCR             STR [0] "Opening item dispute for cycle forward"
```

Example 1–65 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/cycle_forward 14672 2
0 PIN_FLD_RESULT           ENUM [0] 1
0 PIN_FLD_DESCR             STR [0] "Successful"
0 PIN_FLD_RESULTS         ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/dispute/item
219444928718058598 0
0 PIN_FLD_RESULTS         ARRAY [1] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/item/transfer
219444928718060646 0
```

PCM_OP_AR_ITEM_SETTLEMENT

Settles an item that is in dispute.

See the discussion on settling disputed items in *BRM Managing Accounts Receivable*.

Example 1–66 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/cycle_forward 199090 0
0 PIN_FLD_AMOUNT        DECIMAL [0] -3
0 PIN_FLD_CURRENCY          INT [0] 840
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_DESCR             STR [0] "Settlement in full"
```

Example 1–67 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /item/cycle_forward 199090 0
0 PIN_FLD_RESULT           ENUM [0] 3
0 PIN_FLD_DESCR             STR [0] "Successful"
0 PIN_FLD_RESULTS         ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/settlement/item
                                     216964430485980618 0
0 PIN_FLD_RESULTS         ARRAY [1] allocated 1, used 1
1     PIN_FLD_POID          POID [0] 0.0.0.1 /event/billing/item/transfer
216964430485982666 0
```

PCM_OP_AR_ITEM_WRITEOFF

Performs a write-off adjustment of an item.

See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.

Example 1–68 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /item/cycle_forward 199090 0
0 PIN_FLD_PROGRAM_NAME     STR [0] "Customer Center"
0 PIN_FLD_DESCR            STR [0] "abc"
```

Example 1–69 Sample Output Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /item/cycle_forward 199090 0
0 PIN_FLD_RESULT        ENUM [0] 1
0 PIN_FLD_RESULTS      ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID      POID [0] 0.0.0.1 /event/billing/writeoff/item
216964430485982154 0
```

PCM_OP_AR_RESOURCE_AGGREGATION

Calculates the aggregated amount of each resource for an event. If there is an adjustment, dispute, or settlement associated with the event, the opcode also calculates the aggregated dispute, adjustment or settlement amount for each resource. The resource types can include currency resources, noncurrency resources, or both.

Customer Center uses the aggregated amounts to display the balance impact of an event and help the CSR determine how much of each resource for an event is actually available for A/R activities like adjustments.

See the discussion on retrieving details on available resources in *BRM Managing Accounts Receivable*.

Example 1–70 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 17106 0
0 PIN_FLD_EVENTS            ARRAY [0] allocated 1, used 1
1   PIN_FLD_POID             POID [0] 0.0.0.1
                                      /event/billing/product/fee/cycle/cycle_
forward_monthly
                                      16434 0
```

Example 1–71 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 17106 0
0 PIN_FLD_EVENTS            ARRAY [0] allocated 20, used 5
1   PIN_FLD_POID             POID [0] 0.0.0.1
                                      /event/billing/product/fee/cycle/cycle_
forward_monthly
                                      16434 0
1   PIN_FLD_START_T         TSTAMP [0] (1090220219) Sun Jul 18 23:56:59 2004
1   PIN_FLD_END_T           TSTAMP [0] (1090220219) Sun Jul 18 23:56:59 2004
1   PIN_FLD_AMOUNT         DECIMAL [0] 30
1   PIN_FLD_DISCOUNT       DECIMAL [0] 0
0 PIN_FLD_RESULTS                 ARRAY [0] allocated 20, used 7
1   PIN_FLD_RESOURCE_ID             INT [0] 840
1   PIN_FLD_AMOUNT              DECIMAL [0] 30
1   PIN_FLD_DISCOUNT            DECIMAL [0] 0
1   PIN_FLD_ORIG_DISPUTE_AMOUNT DECIMAL [0] -10
1   PIN_FLD_AMOUNT_ADJUSTED     DECIMAL [0] -3
1   PIN_FLD_ADJUSTED            DECIMAL [0] -6
1   PIN_FLD_DISPUTED            DECIMAL [0] -5
1   PIN_FLD_ALLOCATED           DECIMAL [0] 16 (i.e. 30-(3+6+5))
0 PIN_FLD_RESULTS                 ARRAY [1] allocated 20, used 7
1   PIN_FLD_RESOURCE_ID             INT [0] 1000020
1   PIN_FLD_AMOUNT              DECIMAL [0] 20
1   PIN_FLD_DISCOUNT            DECIMAL [0] 0
1   PIN_FLD_ORIG_DISPUTE_AMOUNT DECIMAL [0] 0
1   PIN_FLD_AMOUNT_ADJUSTED     DECIMAL [0] 0
1   PIN_FLD_DISPUTED            DECIMAL [0] 0
1   PIN_FLD_ADJUSTED            DECIMAL [0] 7
1   PIN_FLD_AMOUNT_ADJUSTED     DECIMAL [0] 0
1   PIN_FLD_ALLOCATED           DECIMAL [0] 13 (i.e. 20-(7))
```

PCM_OP_AR_REVERSE_REFUND

Reverses a SEPA Credit Transfer refund from the account balance and reopens the refund items that were previously closed when the refund was recorded.

This opcode is called by the **pin_sepa** utility to reverse a refund transation that is rejected by the bank.

See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.

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.

This opcode accepts an array of written-off items that need to be reversed. If there are no written-off items in the input flist, it calls the PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode to retrieve the items that need to be reversed.

To perform reversal of a written-off bill unit, you specify the POID of the written-off bill unit.

Note: The automatic write-off reversal feature must be enabled for this opcode to perform write-off reversals. See the discussion on enabling automatic write-off reversals during payment collection in *BRM Managing Accounts Receivable*.

See the discussion on writing off debts and reversing write-offs with your custom application in *BRM Managing Accounts Receivable*.

Example 1–72 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 13416 0
0 PIN_FLD_PROGRAM_NAME        STR [0] "DSG-reversal"
0 PIN_FLD_REVERSALS         ARRAY [0] allocated 2, used 2
1     PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/writeoff 13080 1
1     PIN_FLD_FLAGS           INT [0] 1
0 PIN_FLD_REVERSALS         ARRAY [1] allocated 2, used 2
1     PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/writeoff 14488 1
1     PIN_FLD_FLAGS           INT [0] 1
```

Example 1–73 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 13416 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 3, used 2
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/writeoff_reversal
12696 0
1     PIN_FLD_EVENT_OBJ      POID [0] 0.0.0.1 /event/billing/writeoff_reversal/tax
13720 0
1     PIN_FLD_RESULT         ENUM [0] 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 1, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/writeoff_reversal
14232 0
1     PIN_FLD_RESULT         ENUM [0] 0
```

Active Session Manager FM Standard Opcodes

The opcodes in Table 1–8 maintain state information for prepaid telco sessions that are in progress.

Header File

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

Opcode Index

Table 1–8 Active Session Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ASM_CLOSE_ACTIVE_SESSION	Closes, cancels, or deletes an **/active_session** object.	Recommended
PCM_OP_ASM_CREATE_ACTIVE_SESSION	Creates **/active_session** objects.	Recommended
PCM_OP_ASM_FIND_ACTIVE_SESSION	Finds one or more **/active_session** objects.	Recommended
PCM_OP_ASM_UPDATE_ACTIVE_SESSION	Updates information in an existing **/active_session** object.	Recommended

PCM_OP_ASM_CLOSE_ACTIVE_SESSION

Closes, cancels, or deletes **/active_session** objects depending on the value passed in the PIN_FLD_STATUS_FLAGS field:

- **0** specifies to save the object.
- **1** specifies to delete the object.

This opcode is called by PCM_OP_ACT_END_SESSION when ending a prepaid session.

Example 1–74 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gsm
0 PIN_FLD_ACTIVE_SESSION_ID     STR [0] "4085551212-4085557894-109539771-network"
0 PIN_FLD_STATUS_FLAG          ENUM [0] 0
0 PIN_FLD_STATUS                INT [0] 1
```

Example 1–75 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gsm 56486464
11
```

PCM_OP_ASM_CREATE_ACTIVE_SESSION

Creates **/active_session** objects in IMDB Cache memory or, if IMDB Cache is not installed, in the BRM database.

This opcode is called by PCM_OP_ACT_AUTHORIZE when authorizing prepaid telco sessions.

See the discussion on how BRM authorizes users to access prepaid services in *BRM Telco Integration*.

Example 1–76 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_ACCOUNT_OBJ             POID [0] 0.0.0.1 /account 53824 0
0 PIN_FLD_SERVICE_OBJ             POID [0] 0.0.0.1 /service/telco/gsm/telephony
56128 0
0 PIN_FLD_ACTIVE_SESSION_ID        STR [0] "TEL_AUTH_025"
0 PIN_FLD_PROGRAM_NAME             STR [0] "testnap"
0 PIN_FLD_USAGE_TYPE               STR [0] ""
0 PIN_FLD_STATUS                  ENUM [0] 2
0 PIN_FLD_INHERITED_INFO      SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_TELCO_INFO        SUBSTRUCT [0] allocated 20, used 3
2     PIN_FLD_NETWORK_SESSION_ID    STR [0] "TEL_AUTH_025"
2     PIN_FLD_CALLING_FROM          STR [0] "04222549752"
1   PIN_FLD_GSM_INFO          SUBSTRUCT [0] allocated 20, used 3
2     PIN_FLD_NUMBER_OF_UNITS       INT [0] 0
2     PIN_FLD_DIRECTION            ENUM [0] 1
2     PIN_FLD_DIALED_NUMBER         STR [0] "04222642264"
```

Example 1–77 Sample Output Flist

```
0 PIN_FLD_ACCOUNT_OBJ                POID [0] 0.0.0.1 /account 53824 0
0 PIN_FLD_ACTIVE_SESSION_ID           STR [0] "TEL_AUTH_025"
0 PIN_FLD_AMOUNT                  DECIMAL [0] -1
0 PIN_FLD_CREATED_T                TSTAMP [0] (1121964081) Thu Jul 21 09:41:21
2005
0 PIN_FLD_END_T                    TSTAMP [0] (0) <null>
0 PIN_FLD_MOD_T                    TSTAMP [0] (1121964081) Thu Jul 21 09:41:21
2005
0 PIN_FLD_POID                       POID [0] 0.0.0.1 /active_session/telco/gsm
61207 0
0 PIN_FLD_PROGRAM_NAME                STR [0] "testnap"
0 PIN_FLD_READ_ACCESS                 STR [0] ""
0 PIN_FLD_SERVICE_OBJ                POID [0] 0.0.0.1
/service/telco/gsm/telephony 56128 0
0 PIN_FLD_SESSION_ID                  INT [0] 0
0 PIN_FLD_START_T                  TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS                     ENUM [0] 2
0 PIN_FLD_TIMEZONE_ID                 STR [0] ""
0 PIN_FLD_USAGE_TYPE                  STR [0] ""
0 PIN_FLD_WRITE_ACCESS                STR [0] ""
0 PIN_FLD_TELCO_INFO              SUBSTRUCT [0] allocated 20, used 14
1   PIN_FLD_CALLED_NUM_MODIF_MARK     ENUM [0] 0
1   PIN_FLD_CALLED_TO                 STR [0] "04222549752"
1   PIN_FLD_CALLING_FROM              STR [0] "9886193039"
1   PIN_FLD_DESTINATION_NETWORK       STR [0] ""
1   PIN_FLD_ERA_TYPE                  INT [0] 0
1   PIN_FLD_NETWORK_SESSION_ID        STR [0] "TEL_AUTH_025"
1   PIN_FLD_ORIGIN_NETWORK            STR [0] ""
```

```
1   PIN_FLD_PRIMARY_MSID                STR [0] ""
1   PIN_FLD_SECONDARY_MSID              STR [0] ""
1   PIN_FLD_SVC_CODE                    STR [0] ""
1   PIN_FLD_SVC_TYPE                    STR [0] ""
1   PIN_FLD_TERMINATE_CAUSE            ENUM [0] 0
1   PIN_FLD_USAGE_CLASS                 STR [0] ""
1   PIN_FLD_USAGE_TYPE                  STR [0] ""
0 PIN_FLD_GSM_INFO                SUBSTRUCT [0] allocated 20, used 14
1   PIN_FLD_BYTES_IN                    INT [0] 0
1   PIN_FLD_BYTES_OUT                   INT [0] 0
1   PIN_FLD_CALLED_NUM_MODIF_MARK      ENUM [0] 0
1   PIN_FLD_CELL_ID                     STR [0] ""
1   PIN_FLD_DESTINATION_SID             STR [0] ""
1   PIN_FLD_DIALED_NUMBER               STR [0] "04222642264"
1   PIN_FLD_DIRECTION                  ENUM [0] 1
1   PIN_FLD_IMEI                        STR [0] ""
1   PIN_FLD_LOC_AREA_CODE               STR [0] ""
1   PIN_FLD_NUMBER_OF_UNITS             INT [0] 0
1   PIN_FLD_ORIGIN_SID                  STR [0] ""
1   PIN_FLD_QOS_NEGOTIATED             ENUM [0] 0
1   PIN_FLD_QOS_REQUESTED              ENUM [0] 0
1   PIN_FLD_SUB_TRANS_ID                STR [0] ""
```

PCM_OP_ASM_FIND_ACTIVE_SESSION

Finds one or more **/active_session** objects.

By default, this opcode searches for **/active_session** objects based on the following criteria passed in the input flist:

- Active session ID
- Status

Example 1–78 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_ACTIVE_SESSION_ID     STR [0] "4085551212-4085557894-109539771-network"
0 PIN_FLD_STATUS               ENUM [0] 1
```

Example 1–79 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /active_session/telco/gsm 175358 0
0 PIN_FLD_RESULTS          ARRAY [0] allocated 20, used 4
1   PIN_FLD_POID            POID [0] 0.0.0.1 /active_session/telco/gsm 175358 0
```

PCM_OP_ASM_UPDATE_ACTIVE_SESSION

Updates information in an existing **/active_session** object.

This opcode is called by PCM_OP_ACT_UPDATE_SESSION when updating information about a prepaid telco session.

If PCM_OP_ASM_UPDATE_ACTIVE_SESSION is called with PIN_FLD_CREDIT_THRESHOLDS in the input flist, it uses the credit threshold information to update the **/active_session** object.

See the following discussions:

- How BRM updates prepaid sessions in *BRM Telco Integration*
- How BRM creates credit threshold breach notifications in *BRM Telco Integration*

Example 1–80 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gsm -1 0
0 PIN_FLD_ACTIVE_SESSION_ID     STR [0] "4085551212-4085557894-109539771-network"
0 PIN_FLD_PROGRAM_NAME          STR [0] "sample_act"
0 PIN_FLD_START_T            TSTAMP [0] (1095379771) Thu Sep 15 17:09:31 2004
0 PIN_FLD_END_T              TSTAMP [0] (1095380041) Thu Sep 16 17:14:01 2004
0 PIN_FLD_AMOUNT            DECIMAL [0] 25.0
```

Example 1–81 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gsm 1245972
10
```

Balance Monitoring FM Standard Opcodes

The opcodes in Table 1–9 are used for balance monitoring.

Header File

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

Opcode Index

Table 1–9 Balance Monitoring FM Standard Opcodes

Opcode	Description	Use
PCM_OP_MONITOR_ACCOUNT_HIERARCHY	Updates the list of members in hierarchy-type balance monitors. See the discussion on updating hierarchy-type monitors automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_BILLING_HIERARCHY	Updates the list of members in paying responsibility-type balance monitors. See the discussion on updating paying responsibility-type monitors automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_HIERARCHY_CLEANUP	Removes members from hierarchy-type and paying responsibility-type balance monitors. See the discussion on removing members from hierarchy- and paying responsibility-type monitors in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_PROCESS_BILLING_MONITORS	Adds members to paying responsibility-type balance monitors automatically. See the discussion on adding members to newly created balance monitors automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS	Adds members to hierarchy-type balance monitors automatically. See the discussion on adding members to newly created balance monitors automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS	Adds members to subscription-type balance monitors automatically. See the discussion on adding members to newly created balance monitors automatically in *BRM Managing Accounts Receivable*.	Recommended

Table 1–9 (Cont.) Balance Monitoring FM Standard Opcodes

Opcode	Description	Use
PCM_OP_MONITOR_SERVICE_HIERARCHY	Updates the list of members in subscription-type balance monitors. See the discussion on updating subscription-type monitors automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_SETUP_MEMBERS	Adds members to a balance monitor automatically. See the discussion on adding and removing balance monitor members automatically in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_MONITOR_UPDATE_MONITORS	Generates events to indicate that an **/ordered_balgrp** object was created, modified, or deleted. See the discussion on adding a monitor group to a member's /ordered_balgrp object in *BRM Managing Accounts Receivable*.	Recommended

PCM_OP_MONITOR_ACCOUNT_HIERARCHY

Updates the list of members in hierarchy-type balance monitors. When an account hierarchy changes (for example, when an account is added), this opcode is called to add members to any balance monitors associated with the hierarchy.

This opcode is triggered by the following events:

- **/event/notification/service/pre_purchase**
- **/event/group/member**
- **/event/audit/subscription/transfer**

See the discussion on updating hierarchy-type monitors automatically in *BRM Managing Accounts Receivable*.

> **Note:** Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_BILLING_HIERARCHY

Updates the list of members in paying responsibility-type balance monitors. When an account hierarchy changes (for example, when a service is added), this opcode is called to add members to any balance monitors associated with the hierarchy.

This opcode is triggered by the following events:

- **/event/notification/service/pre_purchase**
- **/event/customer/billinfo/modify**
- **/event/notification/bal_grp/modify**

See the discussion on updating paying responsibility-type monitors automatically in *BRM Managing Accounts Receivable*.

Note: Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_HIERARCHY_CLEANUP

Removes members from hierarchy-type and paying responsibility-type balance monitors. When an account hierarchy changes (for example, when accounts are moved to another hierarchy), this opcode is called to delete members from any associated balance monitors.

This opcode is triggered by the following events:

- **/event/customer/billinfo/modify**
- **/event/group/member**
- **/event/notification/bal_grp/modify**

See the discussion on removing members from hierarchy- and paying responsibility-type monitors in *BRM Managing Accounts Receivable*.

> **Note:** Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_PROCESS_BILLING_MONITORS

Adds members to paying responsibility-type balance monitors automatically. This opcode takes as input the parent of a hierarchy group and automatically adds to the balance monitor the following members:

- The parent account and its services
- All *nonpaying* child accounts and their services

This opcode is called by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

Note: Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS

Adds members to hierarchy-type balance monitors automatically. This opcode takes as input the parent of a hierarchy group and automatically adds to the balance monitor the following members:

- The parent account and its services
- All *paying* child accounts and their services
- All *nonpaying* child accounts and their services

This opcode is called by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

> **Note:** Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS

Adds members to subscription-type balance monitors automatically. This opcode takes as input the parent subscription service and automatically adds to the balance monitor the following members:

- The parent subscription service
- All member services

This opcode is called by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

Note: Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_SERVICE_HIERARCHY

Updates the list of members in subscription-type balance monitors. When a subscription group changes, for example, when a member service is added, this opcode is called to add members to any balance monitors associated with the subscription.

This opcode is triggered by the **/event/notification/service/pre_purchase** notification event.

See the discussion on updating subscription-type monitors automatically in *BRM Managing Accounts Receivable*.

Note: Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_SETUP_MEMBERS

Adds members to a balance monitor automatically. This opcode is triggered by the **/event/group/sharing/monitor/create** event.

This opcode is a wrapper opcode that, according to the monitor group type, calls other standard opcodes to add members to a balance monitor. The opcode called depends on the value of the PIN_FLD_TYPE_STR field, listed in Table 1–10, passed in the input flist:

Table 1–10 PIN_FLD_TYPE_STR Values

PIN_FLD_TYPE_STR Value	Monitor Group Type	Opcode Called
H_CE	Hierarchy	PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS
PR_CE	Paying responsibility	PCM_OP_MONITOR_PROCESS_BILLING_MONITORS
SUB_CE	Subscription	PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS

See the discussion on adding and removing balance monitor members automatically in *BRM Managing Accounts Receivable*.

Note: Although this opcode uses Required transaction handling, it never opens a separate transaction. It can be called only through the event notification feature and requires that a transaction already be open. The opcode errors out if a transaction is not already open.

PCM_OP_MONITOR_UPDATE_MONITORS

Generates the following events to indicate that an **/ordered_balgrp** object was created, modified, or deleted:

- When an **/ordered_balgrp** object is created or modified, generates an **/event/billing/monitor/update** event.
- When an **/ordered_balgrp** object is deleted, generates an **/event/billing/monitor/delete** event.

This opcode is called directly by the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See the discussion on adding a monitor group to a member's /ordered_balgrp object in *BRM Managing Accounts Receivable*.

Example 1–82 Sample Input Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /account 89457
0 PIN_FLD_SERVICE_OBJ            POID [0] 0.0.0.1 /service/ip/gprs 3974 0
0 PIN_FLD_ORDERED_BALGRP_OBJ     POID [0] 0.0.0.1 /ordered_balgrp 121
0 PIN_FLD_ACTION                  STR [0] "Modify"
0 PIN_FLD_MONITOR_IMPACTS       ARRAY [0] allocated 1, used 1
1    PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group/monitor 3421 1
```

Example 1–83 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /event/billing/monitor/update
5874592312
```

Balance FM Policy Opcodes

Use the policy opcodes in Table 1–11 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.

Header File

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

Policy Opcode Index

Table 1–11 Balance FM Policy Opcodes

Opcode	Description	Use
PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS	Creates the offer profile threshold notification for both in-session and out-of-session charging. This policy opcode is called by the PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS opcode and the PCM_OP_RESERVE_EXTEND opcode. See the discussion on the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	After a balance is adjusted for a service that uses a custom life cycle, triggers any required service state change and updates the state expiration date in the **/service** object. By default, this policy opcode supports the sample prepaid service life cycle. You can customize it to support other custom service life cycles. See the discussion on managing service life cycles in *BRM Managing Customers*.	Recommended
PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC	Can be customized to provide a custom algorithm for selecting the default balance group of a bill unit and the default service of the default balance group. See the discussion on specifying the default balance group of a bill unit in *BRM Managing Accounts Receivable* and on specifying the default service of the default balance group in *BRM Managing Accounts Receivable*.	Recommended

PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS

Creates the offer profile threshold notification for both in-session and out-of-session charging.

When you start your BRM system, BRM caches the set of unique resource IDs that you have configured in your offer profiles. This policy opcode processes the threshold calculation logic only if the resource ID is one of the entries in its cache.

This policy opcode is called by the PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS opcode and the PCM_OP_RESERVE_EXTEND opcode.

See the discussion on the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode in *BRM Setting Up Pricing and Rating*.

PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE

By default, after a balance is adjusted for a service that uses a custom life cycle, 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:
   - 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 the PCM_OP_SEARCH opcode 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 the PCM_OP_BAL_GET_BALANCES opcode to get the sum of the balances in the service balance group.
4. Calls the PCM_OP_CUST_UPDATE_SERVICES opcode 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, this policy opcode supports the sample prepaid service life cycle. You can customize it to support other custom service life cycles.

This policy opcode is called by the OP_BAL_APPLY_MULTI_BAL_IMPACTS opcode.

See the discussion on managing service life cycles in *BRM Managing Customers*.

PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC

Allows customization during selection of the default balance group of a bill unit and the default service of the default balance group.

This opcode is not called by any opcode.

See the discussion on specifying the default balance group of a bill unit in *BRM Managing Accounts Receivable* and Specifying the default service of the default balance group in *BRM Managing Accounts Receivable*.

Example 1–84 Sample Input Flist

```
0 PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 151504 0
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /billinfo 149040
0 PIN_FLD_FLAGS                     INT [0] 4
```

Example 1–85 Sample Output Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /billinfo 149040
0 PIN_FLD_RESULTS                   ARRAY [0] allocated 4, used 4
1    PIN_FLD_BAL_GRP_OBJ            POID [0] 0.0.0.1 /balance_group 22661 0
1    PIN_FLD_SERVICE_OBJ            POID [0] 0.0.0.1 /service/ip 21125 8
```

Balance FM Standard Opcodes

The opcodes in Table 1–12 adjust account balances.

Header File

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

Opcode Index

Table 1–12 Balance FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BAL_APPLY_MONITOR_IMPACTS	Updates the balances of monitor groups and performs threshold checking. This opcode is used for balance monitoring. See the discussion on updating monitor balances and sending credit limit or threshold breach notifications in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_CHANGE_VALIDITY	Changes a sub-balance's validity period. See the discussion on modifying a sub-balance in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC	Returns the balance groups and services for all the account's **/billinfo** objects or for a single **/billinfo** object. See the discussion on finding a balance group and service for bill units in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_ACCT_BILLINFO	Returns the main contact information for an account and a list of the account's **/billinfo** objects with the default **/billinfo** marked. See the discussion on finding a bill unit in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_ACCT_MONITORS	Retrieves the list of balance monitors owned by a specified account or service. This opcode is used for balance monitoring. See the discussion on retrieving the balance monitors owned by an account or service in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_BALANCES	Returns the POID of a **/balance_group** object and, optionally, the balances it contains. See the discussion on finding a balance group and its balances in *BRM Managing Accounts Receivable*.	Recommended

Table 1–12 (Cont.) Balance FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BAL_GET_BAL_GRP_AND_SVC	Gets the balance groups and services for a **/billinfo** object. See the discussion on finding a balance group and service for bill units in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_ECE_BALANCES	Gets the real-time balances for a service from ECE.	Recommended
PCM_OP_BAL_GET_MONITOR_BAL	Retrieves the balance for a specified balance monitor. This opcode is used for balance monitoring. See the discussion on retrieving the balances for a monitor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BAL_GET_PREPAID_BALANCES	Retrieves a customer's current reservation balance.	Recommended
PCM_OP_BAL_LOCK_RESERVATION_LIST	Finds and then locks a balance group's **/reservation_list** object.	Recommended

PCM_OP_BAL_APPLY_MONITOR_IMPACTS

Updates the balances of monitor groups and performs threshold checking. When thresholds are crossed, for each monitor group this opcode generates a single notification event for all crossed thresholds. This opcode is used for balance monitoring.

See the discussion on updating monitor balances and sending credit limit or threshold breach notifications in *BRM Managing Accounts Receivable*.

Example 1–86 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1
/event/delayed/session/telco/gsm 22537
0 PIN_FLD_OBJECT                        POID [0] 0.0.0.1 /monitor_queue 245
0 PIN_FLD_MONITOR_IMPACTS              ARRAY [0] allocated 4, used 4
1    PIN_FLD_ACCOUNT_OBJ                POID [0] 0.0.0.1 /account 59967 10
1    PIN_FLD_BAL_GRP_OBJ                POID [0] 0.0.0.1 /balance_group/monitor 89993
0
1    PIN_FLD_AMOUNT                  DECIMAL [0] 18.0
1    PIN_FLD_RESOUCE_ID                  INT [0] 840
0 PIN_FLD_MONITOR_SUB_BAL_IMPACTS      ARRAY [0] allocated 3, used 3
1    PIN_FLD_BAL_GROUP_OBJ              POID [0] 0.0.0.1 /balance_group/monitor 89993
0
1    PIN_FLD_RESOUCE_ID                  INT [0] 840
1    PIN_FLD_SUB_BALANCES               ARRAY [0] allocated 4, used 4
2       PIN_FLD_AMOUNT               DECIMAL [0] 4.0
2       PIN_FLD_VALID_FROM            TSTAMP [0] (1106709786) Tue Jan 25 19:23:06
2005
2       PIN_FLD_VALID_TO              TSTAMP [0] (1111737600) Fri Mar 25 00:00:00
2005
2       PIN_FLD_CONTRIBUTOR_STR          STR [0] "sample string"
```

Example 1–87 Sample Output Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /account 59967 10
```

PCM_OP_BAL_CHANGE_VALIDITY

Changes a sub-balance's validity period.

See the discussion on modifying a sub-balance in *BRM Managing Accounts Receivable*.

Example 1–88 Sample Input Flist

```
0 PIN_FLD_BAL_GRP_OBJ POID   [0] 0.0.0.1 /balance_group 8958 21
0 PIN_FLD_POID POID          [0] 0.0.0.1 /account 11518 0
0 PIN_FLD_PROGRAM_NAME STR   [0] "testnap"
0 PIN_FLD_RESOURCE_ID INT    [0] 840
0 PIN_FLD_ELEMENT_ID INT     [0] 458971904
0 PIN_FLD_SUB_BALANCES ARRAY [0] allocated 3, used 3
1  PIN_FLD_VALID_FROM TSTAMP  [0] (1056348600) Mon Jun 23 00:00:00 2003
1  PIN_FLD_VALID_TO TSTAMP    [0] (1058857200) Tue Jul 22 00:00:00 2003
0 PIN_FLD_SUB_BALANCES ARRAY [1] allocated 3, used 3
1  PIN_FLD_VALID_FROM TSTAMP [0] (1056365200)
1  PIN_FLD_VALID_TO TSTAMP    [0] (1058857200) Tue Jul 22 00:00:00 2003
```

PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC

Returns the balance groups and services for all the account's **/billinfo** objects or for a single **/billinfo** object. You can pass flags to get the balance group name and service login aliases.

See the discussion on finding a balance group and service for bill units in *BRM Managing Accounts Receivable*.

Example 1–89 Sample Input Flist

This sample flist shows the input of the opcode called for an account.

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /account 16496 18
```

This sample flist shows the input of the opcode called for a single **/billinfo** object.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 16496 18
```

Example 1–90 Sample Output Flist

This sample flist shows the output of the opcode called for an account and returning balances for all the **/billinfo** objects for the account:

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /account 16496 18
0 PIN_FLD_RESULTS                ARRAY [0] allocated 3, used 3
1     PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 18800 0
1     PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.0  0 0
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (1)"
0 PIN_FLD_RESULTS                ARRAY [1] allocated 5, used 5
1     PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 17776 0
1     PIN_FLD_NEXT_ITEM_POID_LIST  STR [0] ""
1     PIN_FLD_ITEM_POID_LIST       STR [0] ""
1     PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.1 /service/email 20336 8
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (1)"
0 PIN_FLD_RESULTS                ARRAY [2] allocated 5, used 5
1     PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 19824 0
1     PIN_FLD_NEXT_ITEM_POID_LIST  STR [0] ""
1     PIN_FLD_ITEM_POID_LIST       STR [0] ""
1     PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.1 /service/ip 19184 8
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (2)"
```

This sample flist shows the output of the opcode called for a single **/billinfo** object.

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /billinfo 19568 0
0 PIN_FLD_RESULTS                ARRAY [0] allocated 3, used 3
1     PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 18800 0
1     PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.0  0 0
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (1)"
0 PIN_FLD_RESULTS                ARRAY [1] allocated 5, used 5
1     PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 17776 0
1     PIN_FLD_NEXT_ITEM_POID_LIST  STR [0] ""
1     PIN_FLD_ITEM_POID_LIST       STR [0] ""
1     PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.1 /service/email 20336 8
1     PIN_FLD_BILLINFO_ID          STR [0] "Billinfo (1)"
```

PCM_OP_BAL_GET_ACCT_BILLINFO

Returns the main contact information for an account and a list of the account's **/billinfo** objects with the default **/billinfo** marked.

Customer Center calls this opcode to get contact and billing information for an account.

See the discussion on finding a bill unit in *BRM Managing Accounts Receivable*.

Example 1–91 Sample Input Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /account 16496 0
```

Example 1–92 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /account 16496 18
0 PIN_FLD_NAMEINFO            ARRAY [1] allocated 19, used 19
1     PIN_FLD_ADDRESS           STR [0] "ABC"
1     PIN_FLD_CANON_COMPANY     STR [0] ""
1     PIN_FLD_CANON_COUNTRY     STR [0] "US"
1     PIN_FLD_CITY              STR [0] "Clara"
1     PIN_FLD_COMPANY           STR [0] ""
1     PIN_FLD_CONTACT_TYPE      STR [0] "Account holder"
1     PIN_FLD_COUNTRY           STR [0] "USA"
1     PIN_FLD_EMAIL_ADDR         STR [0] ""
1     PIN_FLD_FIRST_CANON        STR [0] "vidya"
1     PIN_FLD_FIRST_NAME         STR [0] "Vidya"
1     PIN_FLD_LAST_CANON         STR [0] "vidya"
1     PIN_FLD_LAST_NAME          STR [0] "Vidya"
1     PIN_FLD_MIDDLE_CANON       STR [0] ""
1     PIN_FLD_MIDDLE_NAME        STR [0] ""
1     PIN_FLD_SALUTATION         STR [0] "Mr."
1     PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.0  0 0
1     PIN_FLD_STATE              STR [0] "CA"
1     PIN_FLD_TITLE              STR [0] ""
1     PIN_FLD_ZIP                STR [0] "88111"
0 PIN_FLD_BILLINFO             ARRAY [0] allocated 8, used 8
1     PIN_FLD_POID              POID [0] 0.0.0.1 /billinfo 19568 7
1     PIN_FLD_BILL_OBJ          POID [0] 0.0.0.1 /bill 17008 0
1     PIN_FLD_AR_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 19568 1
1     PIN_FLD_LAST_BILL_T     TSTAMP [0] (1097384009) Sat Oct  9 21:53:29 2004
1     PIN_FLD_NEXT_BILL_T     TSTAMP [0] (1099987200) Tue Nov  9 00:00:00 2004
1     PIN_FLD_PAY_TYPE          ENUM [0] 10001
1     PIN_FLD_BILLINFO_ID        STR [0] "Billinfo (1)"
1     PIN_FLD_FLAGS              INT [0] 1
0 PIN_FLD_BILLINFO             ARRAY [1] allocated 8, used 8
1     PIN_FLD_POID              POID [0] 0.0.0.1 /billinfo 18032 6
1     PIN_FLD_BILL_OBJ          POID [0] 0.0.0.1 /bill 20080 0
1     PIN_FLD_AR_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 18032 1
1     PIN_FLD_LAST_BILL_T     TSTAMP [0] (1097384009) Sat Oct  9 21:53:29 2004
1     PIN_FLD_NEXT_BILL_T     TSTAMP [0] (1099987200) Tue Nov  9 00:00:00 2004
1     PIN_FLD_PAY_TYPE          ENUM [0] 10001
1     PIN_FLD_BILLINFO_ID        STR [0] "Billinfo (2)"
1     PIN_FLD_FLAGS              INT [0] 0
```

PCM_OP_BAL_GET_ACCT_MONITORS

Retrieves the list of balance monitors owned by a specified account or service. This opcode is used for balance monitoring.

See the discussion on retrieving the balance monitors owned by an account or service in *BRM Managing Accounts Receivable*.

Example 1–93 Sample Input Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /account 57654 283
```

Example 1–94 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 57654 283
0 PIN_FLD_MONITORS        ARRAY [0] allocated 1, used 1
1    PIN_FLD_BAL_GRP_OBJ   POID [0] 0.0.0.1 /balance_group/monitor 254
```

PCM_OP_BAL_GET_BALANCES

Returns the POID of a **/balance_group** object and, optionally, the balances it contains.

This opcode also returns balances that start on first usage (when they are impacted for the first time) whose validity periods have not yet been set.

If no balance is available, this opcode returns **0**.

See the discussion on finding a balance group and its balances in *BRM Managing Accounts Receivable*.

Example 1–95 Sample Input Flist

```
0 PIN_FLD_BAL_GRP_OBJ          POID [0] 0.0.0.1 /balance_group 175992 4
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /account 172664 0
0 PIN_FLD_BALANCES             ARRAY [*]
1     PIN_FLD_CURRENT_BAL      DECIMAL [0] 0.0
1     PIN_FLD_CREDIT_LIMIT      DECIMAL [0] 0.0
```

PCM_OP_BAL_GET_BAL_GRP_AND_SVC

Gets the balance groups and services for a **/billinfo** object.

See the discussion on finding a balance group and service for bill units in *BRM Managing Accounts Receivable*.

Example 1–96 Sample Input Flist

```
0 PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 151504 0
0 PIN_FLD_POID             POID [0] 0.0.0.1 /billinfo 149040 0
```

PCM_OP_BAL_GET_ECE_BALANCES

Gets the real-time balances for a service from ECE. This opcode can be called by a custom application.

See the discussion on configuring the Connection Manager for real-time balances for a service from ECE in the ECE documentation.

PCM_OP_BAL_GET_MONITOR_BAL

Retrieves the balance for a specified balance monitor. This opcode is used for balance monitoring.

See the discussion on retrieving the balances for a monitor group in *BRM Managing Accounts Receivable*.

Example 1–97 Sample Input Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /balance_group/monitor 254
0 PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 89457
0 PIN_FLD_DATE_BALANCES      ARRAY [0] allocated 1, used 1
1    PIN_FLD_BAL_DATE        TSTAMP [0] (1111737600) Fri Mar 25 00:00:00 2005
```

Example 1–98 Sample Output Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /balance_group/monitor 254
0 PIN_FLD_BALANCES                ARRAY [0] allocated 5, used 5
1    PIN_FLD_CREDIT_LIMIT        DECIMAL [0] 100.0
1    PIN_FLD_CREDIT_FLOOR        DECIMAL [0] 0.0
1    PIN_FLD_CREDIT_THRESHOLDS       INT [0] 95
1    PIN_FLD_CURRENT_BAL         DECIMAL [0] 53.0
1    PIN_FLD_DATE_BALANCES         ARRAY [0] allocated 2, used 2
2       PIN_FLD_BAL_DATE          TSTAMP [0] (1111737600) Fri Mar 25 00:00:00 2005
2       PIN_FLD_CURRENT_BAL       DECIMAL [0] 22.1
```

PCM_OP_BAL_GET_PREPAID_BALANCES

Retrieves a customer's current reservation balance.

Example 1–99 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /account 175992 4
0 PIN_FLD_BAL_GRP_OBJ      POID [0] 0.0.0.1 /balance_group 1423 0
```

Example 1–100 Sample Output Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /account 175992 4
0 PIN_FLD_BILLINFO_OBJ        POID [0] 0.0.0.1 /bill_info 172664 0
0 PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 3215876 11
0 PIN_FLD_RESERVATION_LIST   ARRAY [0] allocated 1, used 1
1   PIN_FLD_POID              POID [0] 0.0.0.1 /reservation_list 2426879
1   PIN_FLD_BALANCES         ARRAY [0] allocated 1, used 1
2     PIN_FLD_AMOUNT        DECIMAL [0] 15.0
```

PCM_OP_BAL_LOCK_RESERVATION_LIST

If the value of the **balance_coordinator** entry in the Connection Manager (CM) **pin.conf** file is **0**, the opcode locks the balance group's **/reservation_list** object. If a reservation_list object cannot be found, the opcode creates one and then locks it. If the value of the balance_coordinator entry in the CM **pin.conf** file is **1**, the opcode locks the **/balance_group** object.

This opcode is called by the Services framework AAA opcodes before processing the incoming authentication, authorization, and accounting (AAA) requests for prepaid usage.

See the discussion on updating and reauthorizing prepaid sessions in *BRM Telco Integration*.

Example 1–101 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /account 1265 10
0 PIN_FLD_BAL_GRP_OBJ     POID [0] 0.0.0.1 /balance_group 21657 11
0 PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.1 /service/telco/gsm/telephony 3546486
```

Example 1–102 Sample Output Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /reservation_list 2426879
```

Base Opcodes

The Base opcodes listed in Table 1–13 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.

> **Note:** Each of the DMs included with BRM uses a different implementation of the base opcodes depending on the DM and the storage system it interacts with. For example, the base opcode PCM_OP_SEARCH is implemented differently for the DM_ORACLE and the DM_LDAP.

The Opcode Index lists opcodes that link to detailed information in the opcode descriptions. The opcode description of each opcode includes links to the opcode's flist specifications.

- For information about LDAP base opcodes, see "LDAP Base Opcodes".
- For information about the Email Data Manager opcodes, see "Email Data Manager Opcodes".
- For information common to all opcodes, see the discussion on calling PCM opcodes in *BRM Developer's Guide*.

Header File

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

Opcode Index

Table 1–13 Base Opcodes

Base Opcodes	Description	Use
PCM_OP_BULK_CREATE_OBJ	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*.	Recommended
PCM_OP_BULK_DELETE_OBJ	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*.	Recommended
PCM_OP_BULK_WRITE_FLDS	Updates fields in a large number of objects of the same type. See the discussion on editing a large number of objects in *BRM Developer's Guide*.	Recommended
PCM_OP_CREATE_OBJ	Creates an object. See the discussion on creating objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DELETE_FLDS	Deletes fields from an object. See the discussion on deleting fields in objects in *BRM Developer's Guide*.	Recommended

Table 1–13 (Cont.) Base Opcodes

Base Opcodes	Description	Use
PCM_OP_DELETE_OBJ	Deletes an object. See the discussion on deleting objects in *BRM Developer's Guide*.	Recommended
PCM_OP_GET_DD	Retrieves the Data Dictionary.	Recommended
PCM_OP_GET_PIN_VIRTUAL_TIME	Retrieves the virtual time that is set in the BRM Connection Manager (CM).	Recommended
PCM_OP_GLOBAL_SEARCH	Searches for objects across multiple BRM database schemas. See the discussion on performing a global search in *BRM Developer's Guide*.	Recommended
PCM_OP_GLOBAL_STEP_END	Ends a global step search. See the discussion on ending a global step search in *BRM Developer's Guide*.	Recommended
PCM_OP_GLOBAL_STEP_NEXT	Receives the next set of global step search results. See the discussion on getting the next set of search results from a global step search in *BRM Developer's Guide*.	Recommended
PCM_OP_GLOBAL_STEP_SEARCH	Step-searches across multiple BRM database schemas. See the discussion on performing a global step search in *BRM Developer's Guide*.	Recommended
PCM_OP_INC_FLDS	Increments fields in an object. See the discussion on incrementing fields in objects in *BRM Developer's Guide*.	Recommended
PCM_OP_READ_FLDS	Reads fields from an object. See the discussion on reading fields in an object *BRM Developer's Guide*.	Recommended
PCM_OP_READ_OBJ	Reads an entire object. See the discussion on reading an entire object in *BRM Developer's Guide*.	Recommended
PCM_OP_SEARCH	Searches for objects in a BRM database schema. See the discussion on performing single-schema searches in *BRM Developer's Guide*.	Recommended
PCM_OP_SET_DD	Modifies the Data Dictionary.	Recommended
PCM_OP_STEP_END	Ends a step search. See the discussion on ending a step search in *BRM Developer's Guide*.	Recommended
PCM_OP_STEP_NEXT	Receives the next set of step-search results. See the discussion on getting the next set of search results from a step search in *BRM Developer's Guide*.	Recommended
PCM_OP_STEP_SEARCH	Step-searches for objects in a BRM database schema. See the discussion on performing single-schema step searches in *BRM Developer's Guide*.	Recommended
PCM_OP_TEST_LOOPBACK	Tests directory server connections.	Recommended

Table 1–13 (Cont.) Base Opcodes

Base Opcodes	Description	Use
PCM_OP_TRANS_ABORT	Aborts an open PCM transaction. See the discussion on cancelling transactions in *BRM Developer's Guide*.	Recommended
PCM_OP_TRANS_COMMIT	Commits an open PCM transaction. See the discussion on committing transactions in *BRM Developer's Guide*.	Recommended
PCM_OP_TRANS_OPEN	Opens a PCM transaction. See the discussion on using transactions in *BRM Developer's Guide*.	Recommended
PCM_OP_TRANS_POL_ABORT	Aborts an open PCM transaction. See the discussion on cancelling transactions in *BRM Developer's Guide*	Recommended
PCM_OP_TRANS_POL_COMMIT	Commits an open PCM transaction. See the discussion on committing transactions in *BRM Developer's Guide*.	Recommended
PCM_OP_TRANS_POL_OPEN	Opens a PCM transaction. See the discussion on using transactions in *BRM Developer's Guide*.	Recommended
PCM_OP_TRANS_POL_PREP_COMMIT	Verifies that an external system can commit a transaction. See the discussion on customizing how to verify the readiness of an external system to commit a transaction opcode in *BRM Developer's Guide*.	Recommended
PCM_OP_WRITE_FLDS	Writes fields to an object. See the discussion on writing fields in objects in *BRM Developer's Guide*.	Recommended

PCM_OP_BULK_CREATE_OBJ

This opcode creates a large number of objects of the same type.

It returns the POID type of the objects created.

See the discussion on creating a large number of objects in *BRM Developer's Guide*.

PCM_OP_BULK_DELETE_OBJ

This opcode deletes a large number of objects of the same type and all the fields in the objects. You specify the conditions the objects must meet in a query in the input flist.

It returns the POID type and the range of POIDs of the deleted objects.

See the discussion on deleting a large number of objects in *BRM Developer's Guide*.

Example 1–103 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 1
0 PIN_FLD_TEMPLATE        STR [0] "delete X for /account where F1 like V1 "
0 PIN_FLD_FLAGS          INT [0]  512
0 PIN_FLD_ARGS            ARRAY [1]
1     PIN_FLD_NAMEINFO      ARRAY [*]
2         PIN_FLD_FIRST_CANON     STR [0] "%"
```

Example 1–104 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 1
0 PIN_FLD_COUNT       INT [0] 3
```

PCM_OP_BULK_WRITE_FLDS

This opcode 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. 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.

See the discussion on editing a large number of objects in *BRM Developer's Guide*.

Example 1–105 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 1
0 PIN_FLD_TEMPLATE        STR [0] "update X for /account where F1 like V1 "
0 PIN_FLD_FLAGS            INT[0]  512
0 PIN_FLD_ARGS            ARRAY [1]
1     PIN_FLD_NAMEINFO      ARRAY [*]
2         PIN_FLD_FIRST_CANON     STR [0] "K%"
0 PIN_FLD_VALUES       ARRAY [0]
1     PIN_FLD_STATUS_FLAGS    INT [0] 1
```

Example 1–106 Sample Output Flist

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /account -1 1
0 PIN_FLD_COUNT       INT [0] 3
```

PCM_OP_CREATE_OBJ

This opcode creates a new object of the type specified on the input flist.

It returns the POID of the object created. If you use the PCM_OPFLG_READ_RESULT flag, it also returns all fields from the created object, including array elements and substructures.

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

PCM_OP_DELETE_FLDS

Deletes arrays or array elements from an object.

Returns the POID of the object from which an element was deleted, including the new revision number.

You must delete at least one array element. Specify the POID of the object from which to delete elements on the input flist. Also specify the array element ID for each element to be deleted. To delete an entire array, put the array on the input flist and use the element ID, PCM_RECID_ALL.

See the discussion on deleting fields in objects in *BRM Developer's Guide*.

PCM_OP_DELETE_OBJ

Deletes a specified object and all its fields.

Returns the POID of the deleted object.

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

PCM_OP_GET_DD

Retrieves the Data Dictionary.

Note: Oracle recommends to use the **pin_deploy** utility to invoke this opcode.

PCM_OP_GET_PIN_VIRTUAL_TIME

Retrieves the virtual time that is set in the BRM Connection Manager (CM).

Important: Use this opcode in test environments only. Do not use it in a production system.

Use this opcode in a test environment when you want to retrieve the virtual time for a custom application that is connected to the CM.

You set the virtual time in the **pin_virtual_time** entry of the CM configuration file (**pin.conf**) by first running the **pin_virtual_time** utility. If the virtual time is not set, this opcode returns the system time.

This opcode takes a dummy account POID as input.

Note: To set the virtual time, see **pin_virtual_time**.

PCM_OP_GLOBAL_SEARCH

To perform a global search, use the PCM_OP_GLOBAL_SEARCH opcode. This opcode searches for objects across multiple BRM database schemas.

This opcode enables a client application to search for objects that meet a set of criteria defined by the client application. Use this opcode when you do not know enough about the target object to specify its database schema. If you know the specific schema to search, use PCM_OP_SEARCH instead.

See the discussion on performing a global search in *BRM Developer's Guide*.

PCM_OP_GLOBAL_STEP_END

Ends global step-searching initiated by the PCM_OP_GLOBAL_STEP_SEARCH opcode.

PCM_OP_GLOBAL_STEP_SEARCH sets the criteria for a step search, sets the size of the results, and initiates the search. See that opcode for details. The PCM_OP_GLOBAL_STEP_NEXT opcode only receives results; it does *not* do a search. This opcode ends the step search, freeing the database cursor and returning any shared memory allocated for the results by the DM.

See the discussion on ending a global step search in *BRM Developer's Guide*.

PCM_OP_GLOBAL_STEP_NEXT

Receives the next set of search results from a step search.

This opcode enables a client application to receive the next set of results from a search initiated by the PCM_OP_GLOBAL_STEP_SEARCH opcode.

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

Step-searches for objects across multiple BRM database schemas. This opcode enables a client application to define search criteria, search for objects using that criteria, and receive a specified number of result sets. This opcode is used for global searches across multiple schemas. If you are searching for an object in a known schema, use PCM_OP_STEP_SEARCH instead.

See the discussion on performing a global step search in *BRM Developer's Guide*.

PCM_OP_INC_FLDS

Increments or decrements one or more fields in an object.

This opcode returns the POID of the object whose fields were updated, including the new revision number. It also returns the revised values of the selected fields, unless the PCM_OPFLG_NO_RESULTS flag is used.

See the discussion on incrementing fields in objects in *BRM Developer's Guide*.

PCM_OP_READ_FLDS

Reads one or more fields in an object.

This opcode allows a client application to read specified fields in an object. Specify the POID of the object along with the list of fields to be read on the input flist. The POID is mandatory while the fields are optional. If there are no fields present, only the POID is read and returned.

This opcode returns the POID of the object from which the fields were read, along with the specified fields and their values.

See the discussion on reading fields in an object in *BRM Developer's Guide*.

PCM_OP_READ_OBJ

Reads an entire object from the database.

Specify the POID of the object to read on the input flist.

The POID of the object and all fields in the object are returned, including array elements and substructures.

See the discussion on reading an entire object in *BRM Developer's Guide*.

PCM_OP_SEARCH

Searches for objects in a single BRM database schema.

This opcode enables a client application to search for objects that meet a set of criteria defined by the client application.

Note: If two objects have an encrypted field that contains the same data encrypted with different keys, a PCM_OP_SEARCH for that value returns only one object.

Important: Use this opcode only to search a single, known database schema. If your BRM implementation uses multiple schemas and you need to search more than one, use the PCM_OP_GLOBAL_SEARCH opcode.

Note: When using the PCM_OP_SEARCH opcode, you can apply the *order by* clause only to the top-level arrays. The *order by* clause cannot be applied to subarrays.

See the discussion on performing single-schema searches in *BRM Developer's Guide*.

PCM_OP_SET_DD

Modifies the Data Dictionary.

Note: Oracle recommends to use the **pin_deploy** utility to invoke this opcode.

PCM_OP_STEP_END

Ends a step search initiated by the PCM_OP_STEP_SEARCH opcode.

This opcode must be used in combination with the PCM_OP_STEP_SEARCH and PCM_OP_STEP_NEXT opcodes to complete the step search cycle. PCM_OP_STEP_SEARCH initiates step searching and gets the first set of PIN_FLD_RESULT elements. PCM_OP_STEP_NEXT retrieves the next specified number of results. PCM_OP_STEP_END ends the step search.

See the discussion on ending a step search in *BRM Developer's Guide*.

PCM_OP_STEP_NEXT

Retrieves the next set of search results from a step search.

This opcode enables a client application to receive the next set of results from a search initiated by PCM_OP_STEP_SEARCH. Results of the search are returned in discrete chunks.

See the discussion on getting the next set of search results from a step search in *BRM Developer's Guide*.

PCM_OP_STEP_SEARCH

Searches for objects in a single BRM database schema.

Important: Use this opcode only to search a single, known database schema. If your BRM implementation uses multiple schemas and you need to search more than one, use the PCM_OP_GLOBAL_STEP_SEARCH opcode.

See the discussion on performing a single-schema step search in *BRM Developer's Guide*.

PCM_OP_TEST_LOOPBACK

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_TRANS_ABORT

Aborts an open PCM transaction.

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

Note: The PIN_FLD_TRANS_ID field in the input flist is reserved for internal use.

PCM_OP_TRANS_COMMIT

Commits an open transaction on a PCM context.

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

Note: The PIN_FLD_TRANS_ID and PIN_FLD_FLAGS fields in the input flist are reserved for internal use.

PCM_OP_TRANS_OPEN

Opens a transaction on a PCM context.

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

Use the following flags to open different types of transactions:

- PCM_TRANS_OPEN_READONLY. See the discussion on read-write transactions in *BRM Developer's Guide*.
- PCM_TRANS_OPEN_READWRITE. See the discussion on read-only transactions *BRM Developer's Guide*.
- PCM_TRANS_OPEN_LOCK_OBJ. See the discussion on transaction with a locked objects in *BRM Developer's Guide*.

The following input flist fields are reserved for internal use:

- PIN_FLD_TRANS_ID
- PIN_FLD_TRANS_TIMEOUT_IN_SECS
- PIN_FLD_FLAGS

PCM_OP_TRANS_POL_ABORT

Aborts an open PCM transaction.

The return flist from PCM_OP_TRANS_POL_OPEN becomes the transaction ID flist; it can contain whatever you want to put in it. That flist then becomes the input to PCM_OP_TRANS_POL_ABORT. The return flist from this opcode is ignored.

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

PCM_OP_TRANS_POL_COMMIT

Commits the current transaction.

The return flist from PCM_OP_TRANS_POL_OPEN becomes the transaction ID flist; it can contain whatever you want to put in it. That flist then becomes the input to PCM_OP_TRANS_POL_COMMIT. The return flist from this opcode is ignored.

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

PCM_OP_TRANS_POL_OPEN

Gets the same flist that PCM_OP_TRANS_OPEN does. The return flist then becomes the transaction ID flist; it can contain whatever you want to put in it. That flist then becomes the input to PCM_OP_TRANS_POL_COMMIT and PCM_OP_TRANS_POL_ABORT. The return flists from those opcodes are ignored.

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

PCM_OP_TRANS_POL_PREP_COMMIT

Enables BRM to confirm the readiness of an external system to commit a transaction.

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

Writes fields in an object.

This opcode allows a client application to set the values of fields in an object. Specify the fields and values to set, along with the POID of the object, on the input flist. You must update at least one field.

Returns the POID of the object whose fields were written, including the new revision number.

See the discussion on writing fields in objects in *BRM Developer's Guide*.

Batch Suspense Manager FM Standard Opcodes

The opcodes listed in Table 1–14 manage batch files for suspended EDRs stored in the BRM database as **/suspended_batch** objects.

Header File

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

Opcode Index

Table 1–14 Batch Suspense Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES	Deletes suspended batches from the BRM database. Available with Suspense Manager. See the discussion on deleting records for suspended batches in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES	Resubmits the batches which have been suspended by the pipeline. See the discussion on resubmitting Suspended Batches in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES	Writes off suspended batches. See the discussion on writing off suspended batches in *BRM Configuring Pipeline Rating and Discounting*.	Recommended

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES

Deletes suspended batches from the BRM database.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on deleting records for suspended batches in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–107 Sample Input Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /admin_action/suspended_
batch/1rec 0 0
0 PIN_FLD_PROGRAM_NAME              STR [0] "TestNap"
0 PIN_FLD_SUSPENDED_BATCH_OBJS      ARRAY [0] allocated 13, used 13
1   PIN_FLD_SUSPENDED_BATCH_OBJ     POID [0] 0.0.0.1 /suspended_batch/telco
15204 0
```

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES

Initiates batch resubmission. During the resubmission process, suspended batches are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to resubmit suspended batches.

See the discussion on resubmitting Suspended Batches in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–108 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /admin_action/suspended_
batch/1rec 0 0
0 PIN_FLD_PROGRAM_NAME                  STR [0] "TestNap"
0 PIN_FLD_BATCH_OVERRIDE_REASONS        STR [0] 1.2, 2.2
0 PIN_FLD_SUSPENDED_BATCH_OBJS          ARRAY [0] allocated 13, used 13
1   PIN_FLD_SUSPENDED_BATCH_OBJ         POID [0] 0.0.0.1 /suspended_batch/telco
12530 0
```

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES

Writes off the batches which are at the "Suspended" stage because of some business rule. The GUI calls this opcode to write off the batches.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on writing off suspended batches in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–109 Sample Input Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /admin_action/suspended_
batch/1rec 0 0
0 PIN_FLD_PROGRAM_NAME              STR [0] "TestNap"
0 PIN_FLD_SUSPENDED_BATCH_OBJS      ARRAY [0] allocated 13, used 13
1   PIN_FLD_SUSPENDED_BATCH_OBJ         POID [0] 0.0.0.1 /suspended_usage/telco
15204 0
```

Billing FM Policy Opcodes

Use the opcodes in Table 1–15 to customize billing and A/R processes.

Header File

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

Opcode Index

Table 1–15 Billing FM Policy Opcodes

Opcode	Description	Use
PCM_OP_BILL_POL_BILL_PRE_COMMIT	Performs modifications to a bill object before it is committed to the BRM database. See the discussion on customizing how to modify a bill object in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_POL_CALC_PYMT_DUE_T	Calculates the due date and the payment collection date of a bill (**/bill** object). See the following discussions: ▪ How BRM calculates payment collection dates in *BRM Configuring and Collecting Payments* ▪ How BRM calculates bill due dates in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_CHECK_SUPPRESSION	Determines whether a bill should be suppressed. See the discussion on how BRM determines whether bills should be suppressed in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_EVENT_SEARCH	Searches for all events associated with an account. See the discussion on finding events associated with an account in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_POL_GET_EVENT_SPECIFIC_DETAILS	Gets event specific details based on the type of the event.	Recommended
PCM_OP_BILL_POL_GET_ITEM_TAG	Assigns bill items to events. See the discussion on setting up real-time rating to assign items based on event attributes in *BRM Configuring and Running Billing*.	Recommended

Table 1–15 (Cont.) Billing FM Policy Opcodes

Opcode	Description	Use
PCM_OP_BILL_POL_GET_PENDING_ITEMS	Selects the pending items from a **/billinfo** object to be included in a bill created by PCM_OP_BILL_MAKE_BILL_NOW. See the discussion on customizing Bill Now in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_POST_BILLING	Allows post-billing processing of an account. See the discussion on suspending billing of closed accounts in *BRM Configuring and Running Billing*.	Recommended
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, allocates balances on open bills and bill items before performing write-off reversals. See the discussion on customizing reversal of payments allocated to written-off accounts in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_POL_SPEC_BILLNO	Assigns default number to the account object in the database. See the discussion on customizing bill numbers in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE	Allows the customization of accounting cycles. See the discussion on customizing accounting cycles in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_VALID_ADJUSTMENT	Validate information to make adjustments against an item. See the discussion on customizing item-level adjustments in *BRM Managing Accounts Receivable*	Recommended
PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL	Validates a given bill object to determine whether BRM can generate a corrective bill for it. See the discussion on corrective Billing in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_POL_VALID_DISPUTE	Validates information to file a dispute against an item. See the discussion on customizing item-level disputes in *BRM Managing Accounts Receivable*.	Recommended

Table 1–15 (Cont.) Billing FM Policy Opcodes

Opcode	Description	Use
PCM_OP_BILL_POL_VALID_SETTLEMENT	Validate information to settle an item which is in dispute. See the discussion on customizing item-level settlements in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_POL_VALID_TRANSFER	Validate information to transfer money from the payment item to the target item. See the discussion on customizing payment transfer validation in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_POL_VALID_WRITEOFF	Validate information to make write-off adjustments against an item. See the discussion on customizing write-off validation in *BRM Managing Accounts Receivable*.	Recommended

PCM_OP_BILL_POL_BILL_PRE_COMMIT

Use this opcode to modify a bill object before it is committed to the database.

This opcode is called by the PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_BILL_NOW, and PCM_OP_MAKE_BILL_ON_DEMAND standard opcodes.

See the discussion on customizing how to modify a bill object in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_CALC_PYMT_DUE_T

Calculates the due date and the payment collection date of a bill (**/bill** object).

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.

This opcode is called by the PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_CORRECTIVE_BILL, and PCM_OP_BILL_MAKE_BILL_NOW opcodes.

When the PCM_OP_BILL_MAKE_CORRECTIVE_BILL opcode calls PCM_OP_BILL_POL_CALC_PAYMT_DUE_T to calculate the due date for a corrective bill, it provides PIN_OBJ_NAME_CORRECTIVE_BILL as the value in the PIN_FLD_NAME field. This value can be used to customize the logic to calculate due dates for corrective bills.

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 the PCM_OP_BILL_MAKE_BILL and PCM_OP_BILL_MAKE_CORRECTIVE_BILL opcodes 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.

See the following discussions:

- How BRM calculates payment collection dates in *BRM Configuring and Collecting Payments*
- How BRM calculates bill due dates in *BRM Configuring and Running Billing*
- How BRM creates a bill in *BRM Configuring and Running Billing*

This opcode does not return any values. Its output flist, however, contains the PIN_FLD_DUE_T value, which the PCM_OP_BILL_MAKE_BILL opcode uses as the due date of the bill.

This opcode is called by the PCM_OP_BILL_MAKE_BILL and PCM_OP_BILL_MAKE_BILL_NOW opcodes.

See the following discussions:

- How BRM calculates payment collection dates in *BRM Configuring and Collecting Payments*
- How BRM calculates bill due dates in *BRM Configuring and Running Billing*
- How BRM creates a bill in *BRM Configuring and Running Billing*

PCM_OP_BILL_POL_CHECK_SUPPRESSION

Determines whether a bill should be suppressed.

Use this opcode to customize exceptions to bill suppressions.

This opcode is called by the PCM_OP_BILL_MAKE_BILL standard opcode.

See the following discussions in *BRM Configuring and Running Billing*:

- Customizing bill suppression exceptions
- How BRM determines whether bills should be suppressed

PCM_OP_BILL_POL_EVENT_SEARCH

Searches for all events associated with an account.

By default, this opcode returns all the events for the account, but discards dispute, adjustment, and settlement events. The opcode can be customized to retrieve all the events for the account and keep the dispute, adjustment, and settlement events.

This opcode is not called by any opcode.

See the discussion on finding events associated with an account in *BRM Managing Accounts Receivable*.

Example 1–110 Sample Input Flist

The following input flist directs BRM to search for up to ten /event/billing/product/fee/cycle/cycle_forward_monthly events with starting times later than September 15, 2004, 11:30 am.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 15486 10
0 PIN_FLD_THRESHOLD       INT [0] 10
0 PIN_FLD_START_T      TSTAMP [0] (1095273000) Wed Sep 15 11:30:00 2004
0 PIN_FLD_EVENT_TYPE      STR [0] "/event/billing/product/fee/cycle/cycle_forward_
monthly"
```

Example 1–111 Sample Output Flist

The following output flist identifies the one event that meets the input flist criteria.

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /account 189638 10
0 PIN_FLD_RESULTS             ARRAY [1] allocated 4, used 4
1   PIN_FLD_THRESHOLD           INT [0] 0
1   PIN_FLD_RESULT             ENUM [0] 1
1   PIN_FLD_DESCR               STR [0] "Success"
1   PIN_FLD_EVENTS            ARRAY [1] allocated 13, used 13
2     PIN_FLD_EVENT_OBJ         POID [0] 0.0.0.1
/event/billing/product/fee/cycle/cycle_
                                      forward_monthly 231319654298218278 0
2     PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.1 /service/telco/gsm/telephony
188710 0
2     PIN_FLD_CREATED_T        TSTAMP [0] (1136149406) Sun Jan 01 13:03:26 2006
2     PIN_FLD_START_T          TSTAMP [0] (1136149402) Sun Jan 01 13:03:22 2006
2     PIN_FLD_END_T            TSTAMP [0] (1136149402) Sun Jan 01 13:03:22 2006
2     PIN_FLD_DESCR               STR [0] ""
2     PIN_FLD_CALLED_TO           STR [0] ""
2     PIN_FLD_UNIT               ENUM [0] 0
2     PIN_FLD_NET_QUANTITY    DECIMAL [0] 1
2     PIN_FLD_FLAGS               INT [0] 1
2     PIN_FLD_BAL_IMPACTS       ARRAY [1] allocated 3, used 3
3       PIN_FLD_RESOURCE_ID       INT [0] 978
3       PIN_FLD_AMOUNT        DECIMAL [0] 50
3       PIN_FLD_DISCOUNT      DECIMAL [0] 0
2     PIN_FLD_BAL_IMPACTS       ARRAY [2] allocated 3, used 3
3       PIN_FLD_RESOURCE_ID       INT [0] 1000095
3       PIN_FLD_AMOUNT        DECIMAL [0] -3600
3       PIN_FLD_DISCOUNT      DECIMAL [0] 0
2     PIN_FLD_BAL_IMPACTS       ARRAY [3] allocated 3, used 3
3       PIN_FLD_RESOURCE_ID       INT [0] 978
3       PIN_FLD_AMOUNT        DECIMAL [0] -5
3       PIN_FLD_DISCOUNT      DECIMAL [0] 0
```

PCM_OP_BILL_POL_GET_EVENT_SPECIFIC_DETAILS

Gets event specific details based on the type of the event.

This opcode is called by the PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT opcode.

See the discussion on setting up real-time rating to assign items based on event attributes in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_GET_ITEM_TAG

Assigns bill items to events based on event attributes. You can customize this policy opcode to use any event attributes to set and return the desired item tag. By default, this opcode returns the item tag passed in on the input flist.

This opcode is called by the PCM_OP_ACT_USAGE opcode.

See the discussion on setting up real-time rating to assign items based on event attributes in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_GET_PENDING_ITEMS

Selects the pending items to be included in a bill created by PCM_OP_BILL_MAKE_BILL_NOW. You can customize this opcode to select only those pending items you want to be used by PCM_OP_BILL_MAKE_BILL_NOW.

This opcode is called by the PCM_OP_BILL_MAKE_BILL_NOW standard opcode.

See the discussion on customizing Bill Now in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_POST_BILLING

This policy opcode allows you to perform custom processing on a bill unit (**/billinfo** object) at the time of billing. The default implementation of this policy opcode suspends billing of closed accounts.

This opcode is called by the PCM_OP_BILL_MAKE_BILL opcode.

See the discussion on suspending billing of closed accounts in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_REVERSE_PAYMENT

Performs optional processing on payments that were applied to written-off accounts, and that must be reversed.

Important: 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 this policy opcode to perform the task.

This opcode is called by the PCM_OP_BILL_REVERSE_PAYMENT standard opcode.

See the discussion on customizing reversal of payments allocated to written-off accounts in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_SPEC_BILLNO

Assigns a default number to a **/bill** object.

This policy allows customization of the bill number. 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.

This opcode is called by the PCM_OP_BILL_MAKE_BILL, PCM_OP_BILL_MAKE_BILL_NOW, and PCM_OP_MAKE_BILL_ON_DEMAND opcodes.

See the discussion on customizing bill numbers in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_SPEC_FUTURE_CYCLE

This opcode allows you to customize accounting cycles. This opcode can be modified to calculate the next and future accounting cycles appropriate for your business policy.

This opcode is called by PCM_OP_BILL_MAKE_BILL, PCM_OP_CUST_SET_BILLINFO, and PCM_OP_BILL_RESUME_BILLING standard opcodes, and the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode.

See the discussion on customizing accounting cycles in *BRM Configuring and Running Billing*.

PCM_OP_BILL_POL_VALID_ADJUSTMENT

Validates information to make adjustments against an item.

This opcode is called by the PCM_OP_AR_ITEM_ADJUSTMENT and PCM_OP_AR_ACCOUNT_ADJUSTMENT standard opcodes.

See the discussion on customizing item-level adjustments in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_VALID_CORRECTIVE_BILL

This policy opcode validates a bill unit (**/billinfo** object) at the time of corrective billing. It performs default policy validations and/or any custom validations that you provide. This opcode is called by the PCM_OP_BILL_MAKE_CORRECTIVE_BILL opcode.

This opcode 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 this opcode or requires the opcode to validate the A/R charges in the input bill.

The opcode 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 1–112 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

PCM_OP_BILL_POL_VALID_DISPUTE

Validates information to file a dispute against an item.

This opcode is called by the PCM_OP_AR_ITEM_DISPUTE standard opcode.

See the discussion on customizing item-level disputes in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_VALID_SETTLEMENT

Validates information to settle an item which is in dispute.

This opcode is called by the PCM_OP_AR_ITEM_SETTLEMENT standard opcode.

See the discussion on customizing item-level settlements in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_VALID_TRANSFER

Validates information to transfer money from the payment item to the target item.

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

This opcode is called by the PCM_OP_BILL_ITEM_TRANSFER standard opcode.

See the discussion on customizing payment transfer validation in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_POL_VALID_WRITEOFF

Validates information to make write-off adjustments against an item.

This opcode is called by the PCM_OP_AR_ITEM_WRITEOFF standard opcode.

See the discussion on customizing write-off validation in *BRM Managing Accounts Receivable*.

Billing FM Standard Opcodes

The opcodes listed in Table 1–16 manage billing and billing group processes, as well as some A/R and payment processes.

Header File

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

Opcode Index

Table 1–16 Billing FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BILL_CREATE_SPONSORED_ITEMS	Creates **/item/sponsor** objects for sponsoring accounts.	Recommended
PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS	Converts amounts from source currency to destination currency. See the discussion on changing currency conversion rates in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES	Supplies a conversion rate for currency conversion. See the discussion on changing currency conversion rates in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_CYCLE_TAX	Calculates tax on deferred taxable amounts. See the discussion on calculating taxes during billing in *BRM Configuring and Running Billing*.	Last Resort
PCM_OP_BILL_DEBIT	Debits or credits a noncurrency resource. See the discussion on applying debits and credits in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_FIND	Searches for information in a **/bill** object given a bill number. See the discussion on finding a bill in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT	Called by Customer Center to retrieve the discount for events of a given bill item.	Limited
PCM_OP_BILL_GROUP_ADD_MEMBER	Adds one or more accounts to an existing account group. See the discussion on adding a member to an account group in *BRM Managing Accounts Receivable*.	Last Resort

Table 1–16 (Cont.) Billing FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BILL_GROUP_CREATE	Creates a new **/group** object. See the discussion on creating an account group in *BRM Managing Accounts Receivable*.	Last Resort
PCM_OP_BILL_GROUP_DELETE	Deletes an existing **/group** object. See the discussion on deleting an account group in *BRM Managing Accounts Receivable*.	Last Resort
PCM_OP_BILL_GROUP_DELETE_MEMBER	Deletes an account from an existing group. See the discussion on deleting a member from an account group in *BRM Managing Accounts Receivable*.	Last Resort
PCM_OP_BILL_GROUP_GET_CHILDREN	Gets child accounts for a given **/group** object. See the discussion on getting a list of child accounts in an account group in *BRM Managing Accounts Receivable*.	Last Resort
PCM_OP_BILL_GROUP_GET_PARENT	Gets the parent account of an account group. See the discussion on finding the parent of an account group in *BRM Managing Accounts Receivable*.	Last Resort
PCM_OP_BILL_GROUP_MOVE_MEMBER	Moves a group member; deletes the group if it is empty, and creates the new group if it does not exist. See the discussion on moving a group member in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_ITEM_EVENT_SEARCH	Searches the **/event** object for details related to a specific item. See the discussion on finding events associated with bill items in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_ITEM_REFUND	Creates a refund item for a **/bill** or **/billinfo** object. See the discussion on managing refunds with your custom application in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_BILL_ITEM_TRANSFER	Transfers money from a source item to a target item. Each transfer can affect multiple target items in a single A/R bill. See the discussion on transferring resources between items in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_BILL_MAKE_BILL	Creates a **/bill** object for an account or balance group. See the discussion on how BRM creates a bill in *BRM Configuring and Running Billing*.	Last Resort

Table 1–16 (Cont.) Billing FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BILL_MAKE_BILL_NOW	Bills a **/billinfo** object immediately from Customer Center. See the discussion on how Bill Now works in *BRM Configuring and Running Billing*.	Last Resort
PCM_OP_BILL_MAKE_BILL_ON_DEMAND	Bills a **/billinfo** object immediately manually. See the discussion on how billing on demand works in *BRM Configuring and Running Billing*.	Last Resort
PCM_OP_BILL_MAKE_CORRECTIVE_BILL	Creates a corrective bill for a **/billinfo** object. See the discussion on corrective Billing in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_MAKE_TRIAL_BILL	Creates a trial invoice and collects revenue assurance data for trial billing. See the discussion on how trial billing works in *BRM Configuring and Running Billing*.	Limited
PCM_OP_BILL_RCV_PAYMENT	Creates a payment item and records that currency has been received. See the discussion on how BRM receives payments in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION	Deactivates manual account suppression immediately or on a specified future date. See the discussion on how BRM ends manual account suppression in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_REVERSE	Opens a closed payment item and remove the credit. See the discussion on how BRM reverses payments in *BRM Configuring and Collecting Payments*.	Last Resort
PCM_OP_BILL_REVERSE_PAYMENT	Reverses a payment item. See the discussion on how BRM reverses payments in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION	Activates manual account suppression immediately or on a specified future date. See the discussion on how BRM suppresses accounts in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_BILL_SET_BILL_SUPPRESSION	Handles manual bill suppression. See the discussion on how BRM suppresses bills *BRM Configuring and Running Billing*.	Recommended

Table 1–16 (Cont.) Billing FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BILL_SET_LIMIT_AND_CR	Sets the credit limit and consumption rules for both currency and noncurrency resources. See the discussion on how BRM handles consumption rules and credit limits in *BRM Managing Customers*.	Recommended
PCM_OP_BILL_TRANSFER_BALANCE	Transfers resources from one balance group to another balance group. See the discussion on transferring resources between balance groups in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_BILL_VIEW_INVOICE	Finds the invoice file for a given bill POID, and returns the contents of the file to the caller. See the discussion on displaying invoices in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_BILL_CREATE_SPONSORED_ITEMS

Creates **/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 the PCM_OP_BILL_POL_GET_PENDING_ITEMS opcode.

This opcode 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 executed 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.

Note: If this opcode is called, billing time discounts and folds are not applied.

PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS

Converts currency amounts from a source currency to a destination currency.

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.

Important: BRM supports conversion only between the euro and EMU currencies. Conversion between two EMU currencies or between any other currencies is not supported.

The conversion rates are specified in the **/config/currency/conversionrates** object.

See the discussion on changing currency conversion rates in *BRM Configuring and Running Billing*.

PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES

Supplies currency conversion rates.

This opcode is called by PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS for conversion rate information for EMU and euro currencies specified in the **/config/currency/conversionrates** object.

It returns the conversion rate, start and end time of the time range for this rate and currency operator information.

See the discussion on changing currency conversion rates in *BRM Configuring and Running Billing*.

PCM_OP_BILL_CYCLE_TAX

Calculates tax on deferred taxable amounts. To calculate taxes during billing, the PCM_OP_RATE_EVENT calls the PCM_OP_BILL_CYCLE_TAX opcode.

This opcode calls the PCM_OP_RATE_TAX_CALC opcode to perform the tax calculation.

See the discussion on calculating taxes during billing in *BRM Configuring and Running Billing*.

PCM_OP_BILL_DEBIT

Debits or credits a noncurrency resource.

Customer Center calls this opcode to debit sub-balances for a specific **/balance_group** object associated with an account.

See the discussion on applying debits and credits in *BRM Managing Accounts Receivable*.

Example 1–113 Sample Input Flist

```
0 PIN_FLD_POID        POID 0] 0.0.0.1 /account 172944 0
0 PIN_FLD_PROGRAM_NAME STR [0] "Customer Center"
0 PIN_FLD_DESCR        STR [0] "test"
0 PIN_FLD_DEBIT     ARRAY [100002] allocated 20, used 1
1  PIN_FLD_BAL_OPERAND DECIMAL [0] 4
```

Example 1–114 Sample Output Flist

```
0 PIN_FLD_POID      POID [0] 0.0.0.1 /account 172944 0
0 PIN_FLD_RESULTS  ARRAY [0] allocated 1, used 1
1  PIN_FLD_POID      POID [0] 0.0.0.1 /event/billing/debit 216823692997627036 0
```

PCM_OP_BILL_FIND

Locates a **/bill** object, given a bill number.

Use this opcode to search for **/bill** objects instead of using the PCM_OP_SEARCH and PCM_OP_STEP_SEARCH opcodes.

> **Note:** This opcode does not perform authentication.

See the discussion on finding a bill in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT

Called by Customer Center to retrieve the discount for events of a given bill item.

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.

The default mode for this opcode is set by the field PIN_FLD_MODE or the EventChargeDiscountMode business parameter.

For each event it retrieves, it calculates the total amount of each resource and the total discount amount of each resource. This encompasses both real-time rating and rating performed by the Pipeline Rating Engine.

This enables Customer Center to display the item charge details in the Event Details panel. See Customer Center Help for information on viewing event details.

PCM_OP_BILL_GROUP_ADD_MEMBER

Adds one or more accounts to an existing account group.

This opcodes adds accounts to an account group for billing purposes, when the accounts' bill units (**/billinfo** objects) are to be set up in a billing hierarchy.

See the discussion on adding a member to an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_CREATE

Creates a new account group for billing purposes.

See the discussion on creating an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_DELETE

Deletes an existing account group.

See the discussion on deleting an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_DELETE_MEMBER

Deletes one or more accounts from an existing account group.

See the discussion on deleting a member from an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_GET_CHILDREN

Gets child accounts of a given account group.

This opcode returns a members list holding the children account POIDs for an account group set up for billing purposes. Specific account fields may be read for each account (for example, account name) by passing the **/account** object fields of interest in the input list along with the POID of the **/group** object. If the input list only contains the **/group** object POID, all the fields in the account table for each child is returned.

See the discussion on getting a list of child accounts in an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_GET_PARENT

Gets the parent account of a given account group.

This opcode retrieves the parent account of a given **/group** object. The input to this opcode is a account group POID. The account POID identifying the group's parent account is returned.

See the discussion on finding the parent of an account group in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_GROUP_MOVE_MEMBER

Moves a member of one group to another.

This opcode is the recommended way to perform this action. It is a wrapper for the other BILL_GROUP opcodes.

See the discussion on moving a group member in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_ITEM_EVENT_SEARCH

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.

See the discussion on finding events associated with bill items in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_ITEM_REFUND

Creates a refund item for a **/bill** or **/billinfo** object.

In calculate-only mode, this opcode returns the refundable amount.

In the regular mode, this opcode returns the refundable amount in the **/item/refund** object.

See the discussion on managing refunds with your custom application in *BRM Configuring and Collecting Payments*.

Example 1–115 Sample Input Flist

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /account 8961 63
0 PIN_FLD_BILLINFO_OBJ POID [0] 0.0.0.1 /billinfo 11393 63
0 PIN_FLD_PROGRAM_NAME   STR [0] "Customer Center"
```

Example 1–116 Sample Output Flist

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /account 8961 63
0 PIN_FLD_ITEM_OBJ   POID [0] NULL poid pointer
0 PIN_FLD_AMOUNT  DECIMAL [0] 0
0 PIN_FLD_RESULT      ENUM [0] 1
```

PCM_OP_BILL_ITEM_TRANSFER

Transfers money from a source item to a target item.

> **Note:** This opcode can accept items from multiple A/R bills and creates one transfer event for each A/R bill.

See the discussion on transferring resources between items in *BRM Managing Accounts Receivable*.

Example 1–117 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /account 106860 0
0 PIN_FLD_ITEM_OBJ      POID [0] 0.0.0.1 /item/adjustment 197020 0
0 PIN_FLD_PROGRAM_NAME   STR [0] "event adjustment"
0 PIN_FLD_SESSION_OBJ   POID [0] 0.0.0.1 /event/billing/adjustment/event 199089 0
0 PIN_FLD_START_T     TSTAMP [0] (1064969203) Tue Sep 30 17:46:43 2003
0 PIN_FLD_END_T       TSTAMP [0] (1064969203) Tue Sep 30 17:46:43 2003
0 PIN_FLD_ITEMS        ARRAY [0] allocated 20, used 4
1  PIN_FLD_POID         POID [0] 0.0.0.1 /item/cycle_forward 109596 0
1  PIN_FLD_BILL_OBJ     POID [0] 0.0.0.1 /bill 106732 0
1  PIN_FLD_AR_BILL_OBJ  POID [0] 0.0.0.1 /bill 106732 0
1  PIN_FLD_AMOUNT     DECIMAL [0] -0.62
```

Example 1–118 Sample Output Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /item/adjustment 197020 0
0 PIN_FLD_RESULT   ENUM [0] 1
0 PIN_FLD_DESCR     STR [0] "Succeeded"
0 PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
1  PIN_FLD_POID     POID [0] 0.0.0.1 /event/billing/item/transfer
216823692997625244 0
```

PCM_OP_BILL_MAKE_BILL

Creates a **/bill** object for a specified **/billinfo** object.

When called by the PCM_OP_BILL_MAKE_TRIAL_BILL opcode during trial invoicing, this opcode checks the value of the PIN_FLD_FLAGS field. If the value is PIN_INV_TYPE_PARENT this opcode calculates the adjusted, disputed, due, received, and writeoff values from each of the subordinate account's **/invoice/trial** objects. The resulting billing totals are passed to the parent of the subordinate accounts.

For more information, see the discussion on how trial invoicing works in *BRM Designing and Generating Invoices*.

See the discussion on how BRM creates a bill in *BRM Configuring and Running Billing*.

PCM_OP_BILL_MAKE_BILL_NOW

Bills a specified **/billinfo** object immediately from Customer Center. If a **/billinfo** object is not specified this opcode creates a **/bill** for each **/billinfo** for the given account.

In addition, the opcode 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 the **/account** object for a sponsor is supplied, the opcode calls PCM_OP_BILL_CREATE_SPONSORED_ITEMS.

If BRM has been configured for delayed billing, the opcode can determine if there are items from the current and next billing cycle and produce two bills.

See the discussions on how Bill Now works in *BRM Configuring and Running Billing* and Configuring Bill Now in *BRM Configuring and Running Billing*.

PCM_OP_BILL_MAKE_BILL_ON_DEMAND

Creates a **/bill** object immediately after a **/billinfo** object is created, or when a deal is purchased.

See the discussion on how billing on demand works in *BRM Configuring and Running Billing*.

PCM_OP_BILL_MAKE_CORRECTIVE_BILL

This opcode allows you to create 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
```

The opcode 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
```

See **pin_make_corrective_bill** utility in *Developer's Reference*.

PCM_OP_BILL_MAKE_TRIAL_BILL

Creates trial invoices and collects revenue assurance data from trial billing.

If you enable trial billing to collect revenue assurance data, this opcode returns the summarized data in the PIN_FLD_REVENUES_ARRAY field.

The fields on the input flist determine whether 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.

See the discussion on how trial billing works in *BRM Configuring and Running Billing*.

Note: If a start date is not provided, this opcode creates trial invoices for all complete billing cycles before the end date that have not been billed. For accounts with skipped billing cycles, it is possible that more than one trial invoice will be created.

Example 1–119 Sample Input Flist

This example shows that this opcode was called with a **start** and **end** date:

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 12345
0 PIN_FLD_PROGRAM_NAME      STR [0] "pin_trial_bill_accts"
0 PIN_FLD_START_T        TSTAMP [0] (8986622000)
0 PIN_FLD_END_T          TSTAMP [0] (8986622324)
0 PIN_FLD_BILLINFO_OBJ     POID [0] 0.0.0.1 /billinfo 58016 0
0 PIN_FLD_CHECK_SPLIT_FLAG  INT [0] 0
0 PIN_FLD_PREINVOICE_MODE    INT [0] 0
```

Example 1–120 Sample Output Flist

This example shows two trial invoice POIDs created for the account:

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /account 12345
0 PIN_FLD_RESULT     ENUM [0] 1 /* Pass or Fail */
0 PIN_FLD_RESULTS   ARRAY [0]
1 PIN_FLD_POID       POID [0] 0.0.0.1 /invoice/trial 11441 0
0 PIN_FLD_RESULTS   ARRAY [1]
1 PIN_FLD_POID        POID [0] 0.0.0.1 /invoice/trial 11243 0
```

PCM_OP_BILL_RCV_PAYMENT

Creates a payment item and records that payment has been received.

This opcode is called by Payment Tool. Before calling this opcode, Payment Tool calls PCM_OP_PYMT_SELECT_ITEMS to identify the list of items to apply this payment to.

When multiple resource voucher top-ups involve a currency resource, this opcode is called by the PCM_OP_PYMT_COLLECT opcode, which passes balance impact information through the PIN_FLD_TOPUP_RESOURCE_INFO substruct in this opcode's input flist.

See the following discussions in *BRM Configuring and Collecting Payments*:

- How BRM performs top-ups
- How BRM receives payments

Example 1–121 Sample Input Flist

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /account 60704 2
0 PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 60704 0
0 PIN_FLD_PROGRAM_NAME  STR [0] "test"
0 PIN_FLD_BILLINFO_OBJ POID [0] 0.0.0.1 /billinfo 58016 0
0 PIN_FLD_CURRENCY      INT [0] 840
0 PIN_FLD_AMOUNT    DECIMAL [0] 100
0 PIN_FLD_PAYMENT SUBSTRUCT [0] allocated 20, used 5
1  PIN_FLD_AMOUNT   DECIMAL [0] 100
1  PIN_FLD_COMMAND     ENUM [0] 0
1  PIN_FLD_PAY_TYPE    ENUM [0] 10001
1  PIN_FLD_CURRENCY     INT [0] 840
1  PIN_FLD_TRANS_ID      STR [0] "P-1111"
```

Example 1–122 Sample Output Flist

```
0 PIN_FLD_POID     POID [0] 0.0.0.1 /account 60704 2
0 PIN_FLD_ITEM_OBJ POID [0] 0.0.0.1 /item/payment 197532 0
0 PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
1  PIN_FLD_POID     POID [0] 0.0.0.1 /event/billing/payment/check
216823692997626780 0
```

PCM_OP_BILL_REMOVE_ACCOUNT_SUPPRESSION

Deactivates manual account suppression immediately or on a specified future date.

Note: This opcode does not initiate any required provisioning of reactivated services.

See the discussion on how BRM ends manual account suppression in *BRM Configuring and Running Billing*.

PCM_OP_BILL_REVERSE

Opens a closed payment item and removes the credit. This opcode is a wrapper for PCM_OP_BILL_REVERSE_PAYMENT and is called by Payment Tool.

When performing reversals during payment suspense recycling, this opcode 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.

See the discussion on how BRM reverses payments in *BRM Configuring and Collecting Payments*.

PCM_OP_BILL_REVERSE_PAYMENT

Reverses a payment. Opens a payment item, reverses its balance impacts, and changes items to not paid that were previously recorded as paid.

See the discussion on how BRM reverses payments in *BRM Configuring and Collecting Payments*.

PCM_OP_BILL_SET_ACCOUNT_SUPPRESSION

Activates manual account suppression immediately or on a specified future date.

Note: This opcode does not initiate any required provisioning of deactivated services.

See the discussion on how BRM suppresses accounts in *BRM Configuring and Running Billing*.

PCM_OP_BILL_SET_BILL_SUPPRESSION

Handles manual bill suppression.

See the discussion on how BRM suppresses bills in *BRM Configuring and Running Billing*.

PCM_OP_BILL_SET_LIMIT_AND_CR

Sets the credit limit and consumption rules for both currency and noncurrency resources.

By default, this opcode sets or changes the credit limit and consumption rules in the account-level **/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 on the input flist.

If balance monitoring is enabled, this opcode validates a balance monitor by checking whether a credit limit or threshold is crossed when the credit limits or thresholds are added or changed for the balance monitor.

See the following discussions:

- Balance monitoring in *BRM Managing Accounts Receivable*
- How BRM handles consumption rules and credit limits in *BRM Managing Customers*

Example 1–123 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 172944 13
0 PIN_FLD_DESCR             STR [0] ""
0 PIN_FLD_PROGRAM_NAME      STR [0] "Customer Center"
0 PIN_FLD_LIMIT           ARRAY [840] allocated 20, used 2
1  PIN_FLD_CREDIT_FLOOR DECIMAL [0] 10
1  PIN_FLD_CREDIT_LIMIT DECIMAL [0] NULL
0 PIN_FLD_RULES           ARRAY [840] allocated 20, used 1
1  PIN_FLD_CONSUMPTION_RULE     [0] 5
```

Example 1–124 Sample Output Flist

```
0 PIN_FLD_POID      POID [0] 0.0.0.1 /account 172944 13
0 PIN_FLD_RESULTS  ARRAY [0] allocated 1, used 1
1  PIN_FLD_POID      POID [0] 0.0.0.1 /event/billing/limit 216823692997625500 0
```

Flags

- 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 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.

PCM_OP_BILL_TRANSFER_BALANCE

Transfers resources from one balance group to another balance group.

For example, use this opcode to transfer funds from one prepaid calling card (account) to another.

See the discussion on transferring resources between balance groups in *BRM Managing Accounts Receivable*.

PCM_OP_BILL_VIEW_INVOICE

Note: This opcode will be deleted in a future release. It remains temporarily in BRM for backward compatibility. Use PCM_OP_INV_VIEW_INVOICE instead.

Retrieves a formatted invoice from the database. It uses the value in the PIN_FLD_BILL_NO input field to search for the bill object in the **bill_t** table. If the opcode cannot find the bill in **bill_t**, it searched the **history_bills_t** table.

Channel FM Standard Opcodes

The opcodes listed in Table 1–17 are used to propagate object changes from BRM to a directory server.

Header File

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

Opcode Index

Table 1–17 Channel FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CHANNEL_PUSH	Creates **/channel_event** objects whenever an **/account** or **/service** object changes in BRM. See the discussion on understanding the channel framework in *BRM LDAP Manager*.	Recommended
PCM_OP_CHANNEL_SYNC	Propagates **/channel_event** objects to the LDAP Server. See the discussion on understanding the channel framework in *BRM LDAP Manager*.	Limited

PCM_OP_CHANNEL_PUSH

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.

See the following discussions in *BRM LDAP Manager*:

- Understanding the channel framework
- Configuring event notification for LDAP Manager

PCM_OP_CHANNEL_SYNC

Propagates **/channel_event** objects to the LDAP Server.

In previous releases, the **in_channel_export** utility called this opcode to publish batches of channel events to the LDAP database. Now, **pin_channel_export** uses the channel family ID to determine the LDAP database to which the events are published.

Because it is no longer called by **pin_channel_export**, PCM_OP_CHANNEL_SYNC is not recommended, but it can be called by custom applications or for testing purposes.

See the discussion on understanding the channel framework in *BRM LDAP Manager*.

Collections Manager FM Policy Opcodes

Use the opcodes listed in Table 1–18 to customize collections features.

Header File

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

Opcode Index

Table 1–18 Collections Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES	Applies finance charges. See the discussion on applying finance charges in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES	Applies late fees. See the discussion on applying late fees in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT	Can be modified to assign accounts automatically to collections agents. See the discussion on assigning bill units automatically in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_ASSIGN_DCA	When multiple collections agencies are configured, enables automation of the logic to select a debt collections agency (DCA). See the discussion on assigning bill units to debt collections agency in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_CALC_DUE_DATE	Can be customized to set the due date for the actions to be taken for account's bill unit when the bill unit is entered into the collection process. See the discussion on configuring how Collections Manager determines dates in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION	Can be modified to execute custom collections actions. See the discussion on performing custom collections actions in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO	Can be modified to add functionality associated with an account leaving a collections scenario. See the discussion on performing custom actions when a bill unit leaves collections in *BRM Collections Manager*.	Recommended

Table 1–18 (Cont.) Collections Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS	Can be customized to override the action target setting in the **/config/collections_actions** object. See the discussion on customizing to which collection group members to apply collections actions in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS	Takes the collections scenario as input, reads the **/config/collections/scenario_params** object for additional parameters, and validates whether the scenario is valid for the bill unit in collections. See the discussion on getting all valid collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY	When the payment milestone is not met, this policy opcode does the following: ■ Sets the current action status to **Completed**. ■ Calls PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY, which cancels all the outstanding payment milestones and reschedules the collections scenario actions to start from the following day and updates the scenario object.	Recommended
PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT	Supports payment for credit card or debit card accounts when the collections action type is **promise_to_pay** or **collect_payment** and the action is set to **initiate payment**.	Recommended
PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY	Provides additional validation checks before the promise-to-pay action is called for a bill unit (**/billinfo** object). See the discussion on creating a promise-to-pay action in *BRM Collections Manager*.	Recommended

Table 1–18 (Cont.) Collections Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA	Allows customization of dunning letter data before it is stored in the database. See the discussion on customizing dunning letters in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO	Allows customization to include additional attributes to determine whether bill units (**/billinfo** objects) enter or exit collections and perform collections actions. See the discussion on executing automatic collections actions in*BRM Collections Manager*.	
PCM_OP_COLLECTIONS_POL_SELECT_PROFILE	Gets the profile object for the account. See the discussion on mapping bill units to collections profiles in *BRM Collections Manager*.	Recommended

PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES

Applies collections-related finance charges.

You can customize how a finance charge is calculated or add functionality. For example, you can customize PCM_OP_COLLECTIONS_POL_APPLY_FINANCE_CHARGES to calculate the finance charge from the customer's average daily balance rather than the current balance.

This opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode.

See the discussion on applying finance charges in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_APPLY_LATE_FEES

Applies collections-related late fees.

You can customize the opcode to change the way the late fee is calculated or to add additional functionality. For example, you could customize this policy opcode to calculate a percentage-based late fee from the customer's average daily balance rather than the current balance.

This opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION and PCM_OP_COLLECTIONS_CONFIG_SET_ACTION opcodes.

See the discussion on applying late fees in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_ASSIGN_AGENT

This opcode can be modified to change the way account bill units are assigned to collections agents. By default, it is an empty hook.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO and PCM_OP_COLLECTIONS_PROCESS_ACCOUNT standard opcodes.

See the discussion on assigning bill units automatically in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_ASSIGN_DCA

This opcode can be modified to automate the logic of selecting a debt collections agent (DCA) when multiple DCAs are configured. By default, it is an empty hook.

This opcode is called when the **Refer to outside agency** type system collections action is run.

See the discussion on assigning bill units to debt collections agency in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_CALC_DUE_DATE

This opcode can be customized to set the due date for collections actions.

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.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO standard opcode.

PCM_OP_COLLECTIONS_POL_EXEC_POLICY_ACTION

This policy opcode can be modified to perform custom collections actions. For example, you can create an action that sends SMS text messages to the customer's wireless phone. By default, it is an empty hook.

This opcode is called by the PCM_OP_COLLECTIONS_CONFIG_SET_ACTION opcode.

See the discussion on performing custom collections actions in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_EXIT_SCENARIO

Can be customized to perform cleanup or other tasks when a bill unit leaves a collections scenario. For example, you may want to modify customer credit score when bill units exit collections. By default, it is an empty hook.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO, and PCM_OP_COLLECTIONS_PROCESS_ACCOUNT standard opcodes.

See the discussion on performing custom actions when a bill unit leaves collections in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS

This opcode can be customized to override the action target setting in the **/config/collections_actions** object. By default, this policy opcode is an empty hook.

You can customize the PCM_OP_COLLECTIONS_POL_GET_GROUP_TARGET_ACTIONS policy opcode to use additional attributes for deciding to which bill units to apply a collections action:

- To the individual bill unit only
- To the parent and all child bill units in the collections sharing group
- To all child bill units in the collections sharing group

This opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION opcode.

See the discussion on customizing to which collections group members to apply collections actions in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS

Determines if a scenario is valid or not for the current bill unit.

This opcode takes the scenario as input, reads the **/config/collections/scenario_params** object for additional parameters, and validates whether the scenario is valid for the bill unit in collections. If this opcode fails, it validates the next scenario.

This opcode is called from PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS standard opcode.

PCM_OP_COLLECTIONS_POL_HANDLE_BREACH_PROMISE_TO_PAY

This opcode is called when the promised payment milestone is not met. The default implementation cancels all the outstanding payment milestone actions and reschedules the collections scenario actions to start from the following day and updates the scenario object.

PCM_OP_COLLECTIONS_POL_INITIATE_PAYMENT

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.

This opcode is called by the PCM_OP_COLLECTIONS_TAKE_ACTION standard opcode.

PCM_OP_COLLECTIONS_POL_INVOKE_PROMISE_TO_PAY

Allows you to perform additional validation before invoking promise obligation for a bill unit. For example, if you do not want to issue a promise-to-pay agreement that goes beyond 30 days. By default, it is an empty policy hook.

This opcode is called by the PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY standard opcode.

PCM_OP_COLLECTIONS_POL_PREP_DUNNING_DATA

Allows customization of dunning letter data before it is stored in the database. For example, you might want to enrich the standard data with additional information. You can include the date on which the account will be inactivated if payment is not received.

This opcode is called by the PCM_OP_COLLECTIONS_SET_DUNNING_LETTER standard opcode.

See the discussion on customizing dunning letters in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_POL_PROCESS_BILLINFO

Important: The source code for this policy opcode is not shipped with BRM.

Allows you to include additional criteria to determine whether bill units (**/billinfo** objects) should enter and exit collections. Based on the status flag in the input flist, this policy opcode identifies if it is being called during entry or exit.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO opcode.

PCM_OP_COLLECTIONS_POL_SELECT_PROFILE

Maps account bill units to collections profiles. You can customize the opcode to group bill units into collections profiles based on any criteria you choose. For example, you could create profiles based on credit scores.

Before customization, this policy opcode maps all bill units to the default collections profile.

> **Important:** The default profile uses US Dollars for the currency. To use a different currency for collections, you must edit this opcode.

This opcode is called by the PCM_OP_COLLECTIONS_PROCESS_BILLINFO and PCM_OP_COLLECTIONS_PROCESS_ACCOUNT standard opcodes.

See the discussion on mapping bill units to collections profiles in *BRM Collections Manager*.

Collections Manager FM Standard Opcodes

The opcodes in Table 1–19 identify account bill units (**/billinfo** objects) with overdue balances and manage activities to collect those balances.

Header File

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

Opcode Index

Table 1–19 Collections Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_ADD_ACTION	Adds an action to a collections scenario of a bill unit (**/billinfo** object). See the discussion on adding actions to a collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_ASSIGN_AGENT	Assigns a bill unit to an agent. See the discussion on assigning bill units to a collections agent in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS	Calculates aging buckets for a bill unit. See the discussion on retrieving aging buckets information in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION	Deletes an existing collections configuration action. See the discussion on deleting an existing collections action in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE	Deletes an existing collections configuration profile. See the discussion on deleting an existing collections profile in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO	Deletes an existing collections configuration scenario. See the discussion on deleting an existing collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS	Gets a list of all currently defined collections configuration actions. See the discussion on getting all currently defined collections actions in *BRM Collections Manager*.	Recommended

Table 1–19 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES	Gets a list of currently defined collections configuration profiles. See the discussion on getting all currently defined collections profiles in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS	Gets a list of collections configuration scenarios and associated profiles in the current brand. See the discussion on getting all currently defined collections scenarios in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL	Gets details of a selected collections configuration scenario. See the discussion on getting details of a collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES	Gets a list of message templates for the current brand. See the discussion on getting a list of message templates in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_SET_ACTION	Creates or updates a collections configuration action. See the discussion on creating or updating collections actions in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE	Creates or updates a collections configuration profile. See the discussion on creating or updating collections profiles in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_EXEMPT_BILLINFO	Exempts a bill unit from collections. See the discussion on exempting bill units from collections in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GET_ACTION_HISTORY	Gets history information for a collections action. See the discussion on retrieving collections action history information in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS	Gets a list of collections actions assigned to agents. See the discussion on retrieving a list of collections actions in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GET_BILLINFOS	Gets a list of bill units that are in collections. See the discussion on retrieving a list of bill units in collections in *BRM Collections Manager*.	Recommended

Table 1–19 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_GET_DUNNING_LETTER	Gets a formatted dunning letter. See the discussion on retrieving dunning letters in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL	Gets scenario details for a bill unit. See the discussion on retrieving scenario information in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS	Evaluates 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 lists the valid scenarios for the bill unit. See the discussion on getting all valid collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER	Adds members to the collections group. Validations are performed to check if the newly added member is a subordinate account. The subordinate accounts cannot be added as members of a collections group. If the new member is already in the collections group, it is ignored.	Recommended
PCM_OP_COLLECTIONS_GROUP_CREATE	Creates the collections group.	Recommended
PCM_OP_COLLECTIONS_GROUP_DELETE	Deletes the collections group.	Recommended
PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER	Deletes members from the collections group.	Recommended
PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO	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.	Recommended
PCM_OP_COLLECTIONS_GROUP_MODIFY	Modifies the member name or the parent in a collections group.	Recommended
PCM_OP_COLLECTIONS_GROUP_SET_PARENT	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.	Recommended
PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY	For a bill unit that is already in collections, calls the promise-to-pay agreement. See the discussion on creating a promise-to-pay action in *BRM Collections Manager*.	Recommended

Table 1–19 (Cont.) Collections Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_COLLECTIONS_PROCESS_BILLINFO	Determines whether bill units enter or exit collections and performs collections actions. See the discussion on executing automatic collections action in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_RESCHEDULE_ACTION	Reschedules a collections action for a bill unit. See the discussion on rescheduling an action scheduled for a bill unit in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_REPLACE_SCENARIO	Replaces the existing collections scenario for a bill unit with a new collections scenario. See the discussion on replacing a collections scenario in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY	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 the discussion on revoking a promise-to-pay action in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_SET_ACTION_STATUS	Changes status of an action. See the discussion on changing the status of a collections action in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_SET_DUNNING_LETTER	Gathers data for system-generated dunning letters. See the discussion on gathering and storing data for dunning letters in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER	Prepares an invoice reminder message. See the discussion on preparing invoice reminders in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_TAKE_ACTION	Executes pending actions for a bill unit. See the discussion on executing pending actions for a bill unit in *BRM Collections Manager*.	Recommended
PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS	Updates the amount and payment details for **Promise to Pay** and **Collect Payment** actions. See the discussion on updating amount and payment details in *BRM Collections Manager*.	Recommended

PCM_OP_COLLECTIONS_ADD_ACTION

Adds collections actions to a collections scenario of a bill unit. This opcode is called by Collections Center when a CSR inserts the actions into the collections scenario.

See the discussion on adding actions to a collections scenario in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_ASSIGN_AGENT

Assigns bill units (**/billinfo** objects) to a collections agent. This opcode is called by Collections Center when a collections manager assigns a bill unit to a particular agent.

See the discussion on assigning bill units to a collections agent in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CALC_AGING_BUCKETS

Gets aging bucket details for a bill unit (**/billinfo** object). This opcode is called by Collections Center when a CSR displays the distribution of a bill unit's overdue balance over a number of aging buckets.

See the discussion on retrieving aging buckets information in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_DELETE_ACTION

Deletes an existing collections action. This opcode is called by Collections Configuration when a user deletes the collections action.

See the discussion on deleting an existing collections action in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_DELETE_PROFILE

Deletes an existing collections profile. This opcode is called by Collections Configuration when a user deletes the collections profile.

See the discussion on deleting an existing collections profile in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_DELETE_SCENARIO

Deletes an existing collections scenario. This opcode is called by Collections Configuration when a user deletes the scenario.

See the discussion on deleting an existing collections scenario in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_GET_ACTIONS

Gets a list of all currently defined collections actions. This opcode is called by Collections Configuration to retrieve a list of actions and their definitions.

See the discussion on getting all currently defined collections actions in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_GET_PROFILES

Retrieves a list of currently defined collections profiles. This opcode is called by Collections Configuration to display all currently defined profiles.

See the discussion on getting all currently defined collections profiles in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIOS

Gets a list of all collections scenarios and associated profiles in the current brand. This opcode is called by Collections Configuration to list all currently defined scenarios and profiles.

See the discussion on getting all currently defined collections scenarios in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_GET_SCENARIO_DETAIL

Gets details of a particular collections scenario. This opcode is called by Collections Configuration to display details of the selected scenario.

See the discussion on getting details of a collections scenario in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_GET_TEMPLATES

Gets a list of templates for the current brand. When a user creates or updates a collections configuration action that requires a template, this opcode is called by Collections Configuration to display a list of available templates.

See the discussion on getting a list of message templates in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_SET_ACTION

Creates or updates a collections action. This opcode is called by Collections Configuration when a user creates a new action or modifies an existing action.

See the discussion on creating or updating collections actions in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_CONFIG_SET_PROFILE

Creates or updates a collections profile. This opcode is called by Collections Configuration when a user creates or modifies the profile.

See the discussion on creating or updating collections profiles in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_EXEMPT_BILLINFO

Exempts bill units (**/billinfo** objects) from collections. This opcode is called by Collections Center when a CSR exempts a bill unit from collections.

See the discussion on exempting bill units from collections in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_ACTION_HISTORY

Finds historic information about a particular collections action. This opcode is called by Collections Center to display historic details about when an action was assigned to a collections agent, reassigned, rescheduled, changed status, and so on.

See the discussion on retrieving collections action history information *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_AGENTS_ACTIONS

Retrieves a list of collections actions assigned to collections agents. When collections managers request an overview of the workload for the collections agents they supervise, this opcode is called by Collections Center.

See the discussion on retrieving a list of collections actions in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_BILLINFOS

Gets a list of bill units (**/billinfo** objects) that are in collections. This opcode is called by Collections Center to display the bill units in collections that meet search criteria specified by a CSR.

- To assign a bill unit to an agent, use PCM_OP_COLLECTIONS_ASSIGN_AGENT.
- To exempt a bill unit from collections, use PCM_OP_COLLECTIONS_EXEMPT_BILLINFO.

See the discussion on retrieving a list of bill units in collections in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_DUNNING_LETTER

Creates formatted dunning letters. This opcode is called by the **pin_collections_send_dunning** application to retrieve a dunning letter.

See the discussion on retrieving dunning letters in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_SCENARIO_DETAIL

Gets details of the collections scenario for a bill unit. This opcode is called by Collections Center to display details of the collections actions scheduled for the bill unit by the collections scenario.

See the discussion on retrieving scenario information in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GET_VALID_SCENARIOS

Determines all the eligible collections scenarios for a bill unit by considering the entry amount, entry days, severity of a scenario, and additional configurable parameters. The results are ordered by the entry amount in the descending order and severity in the ascending order.

For each collections scenario in the search result, the opcode calls the PCM_OP_COLLECTIONS_POL_GET_VALID_SCENARIOS policy opcode that uses the scenario as input and returns if the scenario is valid for the bill unit being processed or not. If the scenario is valid, the opcode adds the scenario to the list of valid scenarios, else the opcode evaluates the next scenario.

This opcode is called by Collections Configuration during scenario assignment or replacement to list out the valid scenarios from which you can select a replacement scenario.

See the discussion on getting all valid collections scenarios and scenario assignment or replacement in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_GROUP_ADD_MEMBER

This opcode adds a member to the collections group. The opcode checks if the account is already a member of the group object. The opcode validates if the new member is a subordinate account. Only a subordinate account can be a member of the collections group.

PCM_OP_COLLECTIONS_GROUP_CREATE

This opcode creates 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_DELETE

This opcode deletes the collections group.

PCM_OP_COLLECTIONS_GROUP_DELETE_MEMBER

This opcode deletes a member from the collections group.

PCM_OP_COLLECTIONS_GROUP_GET_BILLINFO

For a given bill unit provides information whether it belongs to the collections group or not. It will also provide the details about the pending receivables for the bill unit.

If the bill unit is a parent of a collections group, the opcode provides the name of the group and its members.

PCM_OP_COLLECTIONS_GROUP_MODIFY

This opcode is used to rename the group, change parent, or replace existing members. While replacing the existing members, the opcode validates all the members and replaces the existing set of group members.

PCM_OP_COLLECTIONS_GROUP_SET_PARENT

This opcode assigns a new collections group owner to a collections group. The opcode validates if the bill unit is an owner of any other collections group. A bill unit can be the owner of only one collections group.

PCM_OP_COLLECTIONS_INVOKE_PROMISE_TO_PAY

Invokes the promise-to-pay agreement for a bill unit which is already in collections.

See the discussion on promise-to-pay in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_PROCESS_BILLINFO

Determines whether bill units (**/billinfo** objects) enter or exit collections and performs collections actions. This opcode is called by the **pin_collections_process** utility.

See the discussion on executing automatic collections actions *BRM Collections Manager*.

PCM_OP_COLLECTIONS_RESCHEDULE_ACTION

Reschedules an action that was scheduled by a bill unit's collections scenario. This opcode is called by Collections Center to reschedule the action to be performed on the bill unit.

See the discussion on rescheduling an action scheduled for a bill unit in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_REPLACE_SCENARIO

Replaces the current collections scenario for a bill unit with a new scenario.

See the discussion on replacing a collections scenario in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_REVOKE_PROMISE_TO_PAY

Revokes or cancels the promise-to-pay agreement for a bill unit.

See the discussion on revoking the promise-to-pay action in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_SET_ACTION_STATUS

Changes the status of a collections action. This opcode is called by Collections Center to update the status of an assigned action.

See the discussion on changing the status of a collections action in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_SET_DUNNING_LETTER

Gathers data for system-generated dunning letters.

See the discussion on gathering and storing data for dunning letters in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_SET_INVOICE_REMINDER

Prepares an invoice reminder message. This message is delivered by means of the Universal Message Store (UMS) framework.

This opcode is called by PCM_OP_COLLECTIONS_PROCESS_BILLINFO if the scenario associated with the bill unit calls for an invoice reminder.

See the discussion on preparing invoice reminders in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_TAKE_ACTION

Executes pending actions for a bill unit. This opcode is called by either PCM_OP_COLLECTIONS_PROCESS_BILLINFO or **pin_deferred_act** utility to execute actions.

See the discussion on executing pending actions for a bill unit in *BRM Collections Manager*.

PCM_OP_COLLECTIONS_UPDATE_ACTION_PAYMENT_DETAILS

Updates the amount and payment mode details for **Promise to Pay** and **Collect Payment** actions.

See the discussion on updating amount and payment details in *BRM Collections Manager*.

Content Manager FM Policy Opcodes

Use the opcodes listed in Table 1–20 to customize how Content Manager processes AAA requests from third-party content providers.

Header File

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

Opcode Index

Table 1–20 Content Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CONTENT_POL_ACCOUNTING	Performs customer authorization and validation checks. See the discussion on returning extended accounting data in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_POL_AUTHORIZE	Performs authorization checks. See the discussion on customizing authorization in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_POL_POST_ACCOUNTING	Returns extended accounting data to the caller. See the discussion on returning extended authorization data in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_POL_POST_AUTHORIZE	Returns extended authorization data to the caller. See the discussion on translating extended events in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_POL_RESOLVE_EVENT_EXTENSIONS	Translates name-value pairs in the input flist to field name-value pairs. See the discussion on returning extended accounting data in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_POL_RESOLVE_USER	Resolves the given request ID to the login of the **/service/content** object. See the discussion on resolving customer logins in *BRM Content Manager*.	Recommended

PCM_OP_CONTENT_POL_ACCOUNTING

Performs customer authorization and validation.

This opcode is called by the PCM_OP_CONTENT_ACCOUNTING standard opcode.

See the discussion on returning extended accounting data in *BRM Content Manager*.

PCM_OP_CONTENT_POL_AUTHORIZE

Authorizes customers to access content.

This opcode is called by the PCM_OP_CONTENT_AUTHORIZE standard opcode.

See the discussion on customizing authorization in *BRM Content Manager*.

PCM_OP_CONTENT_POL_POST_ACCOUNTING

Returns extended data to the caller.

This opcode is called by the PCM_OP_CONTENT_ACCOUNTING standard opcode.

See the discussion on returning extended accounting data in *BRM Content Manager*.

PCM_OP_CONTENT_POL_POST_AUTHORIZE

Returns extended data to the caller.

This opcode is called by the PCM_OP_CONTENT_AUTHORIZE standard opcode.

See the discussion on returning extended authorization data in *BRM Content Manager*.

PCM_OP_CONTENT_POL_RESOLVE_EVENT_EXTENSIONS

Translates name-value pairs in the input flist to field name-value pairs. By default, this opcode returns the POID in the input flist.

This opcode is called by the PCM_OP_CONTENT_AUTHORIZE and PCM_OP_CONTENT_ACCOUNTING standard opcodes.

See the discussion on translating extended events in *BRM Content Manager*.

PCM_OP_CONTENT_POL_RESOLVE_USER

Resolves the given request ID to the customer login of the **/service/content** object.

This opcode is called by the PCM_OP_CONTENT_AUTHENTICATE, PCM_OP_CONTENT_CANCEL_AUTHORIZATION, and PCM_OP_CONTENT_AUTHORIZE standard opcodes.

See the discussion on resolving customer logins in *BRM Content Manager*.

Content Manager FM Standard Opcodes

The opcodes listed in Table 1–21 are used to create, modify, or retrieve Content Manager access lists and process AAA requests from content providers.

Header File

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

Opcode Index

Table 1–21 Content Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CONTENT_ACCOUNTING	Charges the customer for content accessed. See the discussion on charging customers for content usage in*BRM Content Manager*.	Recommended
PCM_OP_CONTENT_AUTHENTICATE	Verifies that a **/service/content** object exists with the given customer login. See the discussion on authenticating customers in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_AUTHORIZE	Authorizes a customer to access content. See the discussion on authorizing customers to access third-party content in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_CANCEL_AUTHORIZATION	Cancels a previous user authorization to access content. See the discussion on canceling existing authorizations in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_FIND	Finds the account object with the given customer ID. See the discussion on finding customer accounts in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_GET_SRVC_FEATURES	Reads the content categories from the specified service object. See the discussion on retrieving an access list in *BRM Content Manager*.	Recommended
PCM_OP_CONTENT_SET_SRVC_FEATURES	Sets the content category list in the specified service object. See the discussion on creating or modifying an access list in *BRM Content Manager*.	Recommended

PCM_OP_CONTENT_ACCOUNTING

Charges customers for third-party content usage.

See the discussion on charging customers for content usage in *BRM Content Manager*.

PCM_OP_CONTENT_AUTHENTICATE

Verifies that a **/service/content** object exists with the given customer login.

See the discussion on authenticating customers in *BRM Content Manager*.

PCM_OP_CONTENT_AUTHORIZE

Authorizes a customer to access third-party content.

See the discussion on authorizing customers to access third-party content in *BRM Content Manager*.

PCM_OP_CONTENT_CANCEL_AUTHORIZATION

Cancels a previous authorization to disable access to content.

See the discussion on canceling existing authorizations in *BRM Content Manager*.

PCM_OP_CONTENT_FIND

Finds the account object that contains the given customer ID. This opcode uses the extended data to resolve the request ID supplied in the login field.

See the discussion on finding customer accounts in *BRM Content Manager*.

PCM_OP_CONTENT_GET_SRVC_FEATURES

Retrieves content categories from **/service/content** objects.

See the discussion on retrieving an access list in *BRM Content Manager*.

PCM_OP_CONTENT_SET_SRVC_FEATURES

Creates or modifies **/service/content** objects.

See the discussion on creating or modifying an access list *BRM Content Manager*.

Context Management Opcodes

The opcodes in Table 1–22 manage the communication between a client application and the BRM database.

Header File

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

Opcode Index

Table 1–22 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	Executes a PCM opcode by passing a copy of the input flist.
PCM_OPREF	Executes a PCM opcode by passing a reference to the input flist.

PCM_CONNECT

This opcode 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.

The routine 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.

> **Important:** In your application, when you open a context and connect to the BRM server, perform all the PCM operations before closing the context. Connections add a significant overhead to the system, which affects performance. Therefore, to improve performance, perform all the operations 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 this routine:

```
PIN_ERR_CLEAR_ERR(&ebuf);

PCM_CONNECT(&ctxp, &db_no, &ebuf);

if (PIN_ERR_IS_ERR(&ebuf)) {

...

}
```

PCM_CONTEXT_CLOSE

This opcode closes a context to the BRM system. The context should be closed once it is no longer needed by an application. This operation 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

This opcode returns nothing.

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

Error-Handling

This routine 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

This opcode 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.

Important: In your application, when you open a context and connect to the BRM server, perform all the PCM operations before closing the context. Connections add a significant overhead to the system, which affects performance. Therefore, to improve performance, do all the operations 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*.

Important: If you have client applications running on the same server as the CM or DM, you still need to use a **TCP/IP** connection for invoking opcodes. You also need to 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 aborted.

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.

See the discussions on adding new client applications and writing a custom FM in *BRM Developer's Guide*.

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 need to 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.

By default, session event logs are written each time a context is opened. For more information on performance implications, see the discussion on turning off session event logging in *BRM System Administrator's Guide*.

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. By default, 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.

By default, 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

This opcode 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

This routine 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

Routine successful; operation is complete.

PIN_ERR_NO_MEM

A memory allocation failed.

PIN_ERR_BAD_LOGIN_RESULT

The login failed.

PCM_OP

This opcode is a wrapper function for all PCM operations. 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 routine. 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,
                int32              opcode,
                int32              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 operation (PCM opcode) to be performed. See the "Base Opcodes" for choices.

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 **(int32) 0**.

in_flistp
A pointer to the input flist. See the individual opcode manual pages for the input flist specifications.

ret_flistpp
A pointer to a pointer for passing back the return flist. See the individual opcode manual pages for the return flist specifications. All operations produce a return flist with at least the PIN_FLD_POID field on it. Other fields on the return flist depend on the operation being performed. 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

This opcode 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

Routine successful; operation is complete.

other codes

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

Error-Handling

This routine 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 this opcode 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,
                int32              opcode,
                int32              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 operation (PCM opcode) to be performed. See the "Base Opcodes" for choices.

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 **(int32) 0**.

in_flistp
A pointer to the input flist. See the individual opcode manual pages for the input flist specifications.

ret_flistpp
A pointer to a pointer for passing back the return flist. See the individual opcode manual pages for the return flist specifications. All operations produce a return flist with at least the PIN_FLD_POID field on it. Other fields on the return flist depend on the operation being performed. 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

This opcode 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

  Routine successful; operation is complete.

- *other codes*

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

Error-Handling

This routine 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.

Customer FM Policy Opcodes

Use the opcodes listed in Table 1–23 to customize the business logic to process account information during customer registration.

Header File

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

Opcode Index

Table 1–23 Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_CANONICALIZE	Searches on localized customer inputs. See the discussion on creating a localized version of BRM in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_POL_COMPARE_PASSWD	Compares service or account passwords. See the discussion on implementing password encryption in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_DECRYPT_PASSWD	Decrypts a clear text password. See the discussion on implementing password encryption in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_ENCRYPT_PASSWD	Checks an account or service password. See the discussion on implementing password encryption in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_EXPIRATION_PASSWD	Calculates and sets the expiration date for the password. See the discussion on customizing password expiration in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_CONFIG	Gets new customer configuration information. See the discussion on sending account information to your application when an account is created in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_DB_LIST	Gets a list of database schemas defined to the system in a multischema environment. See the discussion on getting a list of database schemas in *BRM Managing Customers*.	Recommended

Table 1–23 (Cont.) Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_GET_DB_NO	Selects a database schema to use based on the priority you set (also loads the CM cache with information from the **/config/distribution** storable class). See the discussion on selecting a database schema in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_DEALS	Gets deals available for purchase by the given account or service. See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_INTRO_MSG	Gets registration introductory message. See the discussion on specifying an introductory message in *BRM Managing Customers*	Recommended
PCM_OP_CUST_POL_GET_PLANS	Gets registration pricing plans. See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_POPLIST	Gets registration POP list. See the discussion on returning a point-of-presence (POP) list in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_PRODUCTS	Gets products available for purchase by the given account or service. See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS	Retrieves a list of the plans and deals that an account owns. See the discussion on getting a list of plans and deals that an account owns in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_MODIFY_SERVICE	A hook to perform legacy or external activities during the service modification process.	Recommended
PCM_OP_CUST_POL_POST_COMMIT	Registration hook after transaction commit. See the discussion on creating hooks to external programs in*BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER	A hook after transaction commit on purchasing an add-on plan by a customer. See the discussion on creating hooks to external programs in*BRM Managing Customers*.	Recommended

Table 1–23 (Cont.) Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_PRE_COMMIT	Registration hook before transaction commit. See the discussion on creating hooks to external programs in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PRE_DELETE_PAYINFO	Policy hook to take any necessary actions on customized payment information before the associated **/payinfo** object is deleted. See the discussion on creating hooks to external programs in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PREP_AACINFO	Prepares automatic creation data for validation. See the discussion on customizing automatic account creation (AAC) information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PREP_ACCTINFO	Prepares account data for validation. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_POL_PREP_BILLINFO	Prepares billing information for validation. See the discussion on preparing /billinfo data in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_POL_PREP_INHERITED	Prepares inherited customer data for validation. See the discussion on creating customization interfaces in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_POL_PREP_LIMIT	Prepares credit limit information prior to validation. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_POL_PREP_LOCALE	Prepares locale information for validation. See the discussion on managing and customizing locale information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PREP_LOGIN	Prepares service login data for validation. See the discussion on customizing login names in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PREP_NAMEINFO	Prepares customer contact data for validation. See the discussion on customizing name and address information in *BRM Managing Customers*.	Recommended

Table 1–23 (Cont.) Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_PREP_PASSWD	Prepares password data for validation. See the discussion on creating passwords in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_PREP_PAYINFO	Processes inherited fields and prepares a **/payinfo** object. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Limited
PCM_OP_CUST_POL_PREP_PROFILE	Modifies account data prior to issuing the final call. See the discussion on collecting nonstandard account information in *BRM Managing Customers*.	Limited
PCM_OP_CUST_POL_PREP_STATUS	Prepares status information prior to validation. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Limited
PCM_OP_CUST_POL_PREP_TOPUP	Prepares information used to set up and modify standard top-ups and sponsored top-ups. See the discussion on preparing an account's top-up information in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_CUST_POL_READ_PLAN	Reads a given plan and constructs a tree for the services, deals and products associated with that plan. See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_SET_BRANDINFO	Allows brand names to be changed. See the discussion on changing the brand of an account by using a custom application in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_TAX_CALC	Use custom rates to calculate taxes. See the discussion on using custom tax rates in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_POL_TAX_INIT	Load custom tax data into the cache. See the discussion on using custom tax rates in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_POL_TRANSITION_DEALS	Returns the list of deals available for transition. See the discussion on customizing deal transitions in *BRM Managing Customers*.	Recommended

Table 1–23 (Cont.) Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_TRANSITION_PLANS	Returns the list of plans available for transition. See the discussion on customizing deal transitions in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_AACINFO	Validates automatic creation data. See the discussion on customizing automatic account creation (AAC) information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_ACCTINFO	Validates account information. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_POL_VALID_BILLINFO	Validates billing information See the discussion on validating /billinfo data in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_POL_VALID_LIMIT	Validates credit limit information before it is set in the account. See the discussion on customizing credit limits and resource consumption rules in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_LOCALE	Validates locale information. See the discussion on managing and customizing locale information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_LOGIN	Validates service login data. See the discussion on customizing login names in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_NAMEINFO	Validates customer contact data. See the discussion on customizing name and address information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_PASSWD	Validates account or service password data. See the discussion on creating passwords in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_POL_VALID_PAYINFO	Validates inherited fields for a **/payinfo** object. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Limited
PCM_OP_CUST_POL_VALID_PROFILE	Reviews and validates data prior to creating an object. See the discussion on collecting nonstandard account information in *BRM Managing Customers*.	Limited

Table 1–23 (Cont.) Customer FM Policy Opcodes

Opcode	Description	Use
PCM_OP_CUST_POL_VALID_STATUS	Validates credit limit information before it is set in the account. See the discussion on changing the status of an account, bill unit, or service in *BRM Managing Customers*.	Limited
PCM_OP_CUST_POL_VALID_TAXINFO	Validates the VAT certificate number provided at the time of account creation. See the discussion on validating tax information in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_POL_VALID_TOPUP	Validates information used to set up and modify standard top-ups and sponsored top-ups. See the discussion on validating an account's top-up information in *BRM Configuring and Collecting Payments*.	Recommended

PCM_OP_CUST_POL_CANONICALIZE

Searches for localized (non-English) customer input string fields. The default implementation is the US locale. Canonicalization handles Latin based characters only. You must customize this opcode for other languages.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER, PCM_OP_CUST_SET_NAMEINFO standard opcodes, and the Customer Center search screen.

PCM_OP_CUST_POL_COMPARE_PASSWD

Checks account or service password. This operation takes a cleartext password and an encrypted password and performs a comparison to check if the cleartext password was the source value of the encrypted password.

This opcode is called by the PCM_OP_ACT_VERIFY standard opcode.

See the discussion on implementing password encryption in *BRM Managing Customers*.

PCM_OP_CUST_POL_DECRYPT_PASSWD

This opcode decrypts a clear text password.

This opcode is called by the PCM_OP_WAP_AUTHENTICATE, PCM_OP_TERM_IP_DIALUP_AUTHENTICATE, PCM_OP_CONTENT_AUTHENTICATE, and PCM_OP_TCF_AAA_AUTHENTICATE standard opcodes.

See the discussion on implementing password encryption in *BRM Managing Customers*.

PCM_OP_CUST_POL_ENCRYPT_PASSWD

This opcode encrypts a clear text password based on the type of the POID given and/or the requested encryption algorithm. The binary result is stored as an ASCII-like string to facilitate storage. This opcode determines the kind of service being used from the POID passed in. All encryption requests from IP accounts get clear text encryption (to support CHAP).

This opcode is called by the PCM_OP_CUST_SET_PASSWD standard opcode, and the PCM_OP_CUST_POL_COMPARE_PASSWD policy opcode.

See the discussion on implementing password encryption in *BRM Managing Customers*.

PCM_OP_CUST_POL_EXPIRATION_PASSWD

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

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

To change the default password expiry duration, edit the **passwd_age** entry in the Connection Manager (CM) **pin.conf** file. For example, instead of 90 days you can set the expiration duration to 150 days.

This opcode is called by the PCM_OP_CUST_SET_PASSWD standard opcode.

See the following discussions:

- Setting the default password expiry duration in *BRM System Administrator's Guide*
- Customizing password expiration in *BRM Managing Customers*

Example 1–125 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /account 12177 0
```

Example 1–126 Sample Output Flist

```
0 PIN_FLD_POID                           POID [0] 0.0.0.1 /account 12177 0
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] 1078423200 THU MAR 04 10:00:00 2004
```

PCM_OP_CUST_POL_GET_CONFIG

Gets new customer configuration information. PCM_OP_CUST_POL_GET_CONFIG is called after customer registration has been successfully performed to specify the configuration data that should be returned to the client software.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER standard opcode.

See the discussion on sending account information to your application when an account is created in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_DB_LIST

Gets a list of database schemas defined to the system in a multischema environment.

This opcode is not called by any opcode.

See the discussion on getting a list of database schemas in *BRM Managing Customers*.

Example 1–127 Sample Output Flists

Example output flist for a single-schema environment:

```
0 PIN_FLD_POID   POID [0] 0.0.0.0  0 0
```

Example output flist for a multischema environment:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /config/distribution 21658 0
0 PIN_FLD_CREATED_T    TSTAMP [0] (975089074) Fri Nov 24 10:04:34 2000
0 PIN_FLD_MOD_T        TSTAMP [0] (975089074) Fri Nov 24 10:04:34 2000
0 PIN_FLD_READ_ACCESS     STR [0] "G"
0 PIN_FLD_WRITE_ACCESS    STR [0] "S"
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR           STR [0] "Multi_db entries"
0 PIN_FLD_HOSTNAME        STR [0] "-"
0 PIN_FLD_NAME            STR [0] "Multi_db entries"
0 PIN_FLD_PROGRAM_NAME    STR [0] "-"
0 PIN_FLD_VALUE           STR [0] ""
0 PIN_FLD_VERSION         STR [0] ""
0 PIN_FLD_DISTRIBUTION  ARRAY [0] allocated 7, used 7
1     PIN_FLD_CRITERION       STR [0] ""
1     PIN_FLD_CURR_ACCOUNT_SIZE    INT [0] 100000
1     PIN_FLD_DB_NO           INT [0] 1
1     PIN_FLD_DB_PRIORITY     INT [0] 10
1     PIN_FLD_DB_STATUS      ENUM [0] 1
1     PIN_FLD_DESCR           STR [0] "Multi_db entry 1"
1     PIN_FLD_MAX_ACCOUNT_SIZE    INT [0] 1000000
0 PIN_FLD_DISTRIBUTION  ARRAY [1] allocated 7, used 7
1     PIN_FLD_CRITERION       STR [0] ""
1     PIN_FLD_CURR_ACCOUNT_SIZE    INT [0] 100000
1     PIN_FLD_DB_NO           INT [0] 2
1     PIN_FLD_DB_PRIORITY     INT [0] 10
1     PIN_FLD_DB_STATUS      ENUM [0] 1
1     PIN_FLD_DESCR           STR [0] "Multi_db entry 2"
1     PIN_FLD_MAX_ACCOUNT_SIZE    INT [0] 1000000
```

PCM_OP_CUST_POL_GET_DB_NO

In a multischema system, selects a database schema to use based on the schema priorities set in the *BRM_Home*/**apps/multi_db/config_dist.conf** file.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER standard opcode.

See the discussion on selecting a database schema in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_DEALS

Get a list of deals available for purchase by the given account or service.

This opcode is not called by any opcode.

See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_INTRO_MSG

Get registration introductory message. The type of account and pricing plan selected are passed to the operation on the input flist. This allows 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 allows the introductory message to include HTML, graphics and other complex information.

This opcode is not called by any opcode.

See the discussion on specifying an introductory message in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_PLANS

Retrieves and displays pricing plans based on customer information during registration. Your customer's Automatic Account Creation fields are passed into the operation on the input flist, allowing the opcode to qualify different price plans depending on the values of the fields returned by each customer.

You need to customize this policy opcode to search for and display the plan list by customer types. By default, if you pass in a type-only POID, the opcode retrieves the **new** plans, else it retrieves **addon** plans.

This opcode is not called by any opcode.

See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_POPLIST

Retrieves either the best POP (point-of-presence) for a registering customer to call, or the entire list of POPs so the customer can choose one.

This opcode is not called by any opcode.

See the discussion on returning a point-of-presence (POP) list in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_PRODUCTS

Get a list of products available for purchase by the given account or service. Retrieves a list of products that are available for purchase by the given account or service. The product's PIN_FLD_PERMITTEDS array is checked for valid object types purchasing the product.

This opcode is not called by any opcode.

See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.

PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS

Retrieves a list of plans, deals, or both that an account owns.

This opcode is not called by any opcode.

See the discussion on getting a list of plans and deals that an account owns in *BRM Managing Customers*.

Example 1–128 Sample Input Flist

```
# number of field entries allocated 20, used 1
0 PIN_FLD_POID          POID [0] 0.0.0.1 /account 38298 37
```

Example 1–129 Sample Output Flist

```
# number of field entries allocated 2, used 2
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 14994 0
0 PIN_FLD_PLAN          ARRAY [0] allocated 3, used 3
1     PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.1 /plan 13458 0
1     PIN_FLD_BAL_INFO      ARRAY [0] allocated 2, used 2
2         PIN_FLD_NAME            STR [0] "Balance Group (1)"
2         PIN_FLD_LIMIT         ARRAY [840] allocated 3, used 3
3             PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
3             PIN_FLD_CREDIT_LIMIT DECIMAL [0] 100000
3             PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
1     PIN_FLD_SERVICES      ARRAY [0] allocated 6, used 6
2         PIN_FLD_BAL_INFO_INDEX    INT [0] 1
2         PIN_FLD_SERVICE_ID      STR [0] ""
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip 16274 0
2         PIN_FLD_SUBSCRIPTION_INDEX    INT [0] 0
2         PIN_FLD_BOOLEAN         INT [0] 1
2         PIN_FLD_DEALS         ARRAY [0] allocated 3, used 3
3             PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 8814 0
3             PIN_FLD_BOOLEAN         INT [0] 1
3             PIN_FLD_NODE_LOCATION    STR [0]
"valhalla#18120/1#20040531-175418.634158:valhalla#18120/1#20040531-175418.670167#0
"
```

PCM_OP_CUST_POL_MODIFY_SERVICE

Provides a hook to perform legacy or external activities by using the service information in the input flist during the service modification process.

PCM_OP_CUST_POL_POST_COMMIT

Provides a mechanism to easily insert a trigger to external or legacy systems during customer registration.

The default implementation supports sending a welcome email message to the new customer.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER standard opcode.

See the following discussions in *BRM Managing Customers*:

- Creating hooks to external programs
- Sending welcome messages to customers

PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER

Provides a hook for after the transaction of purchasing an add-on plan by a customer have been completed and committed. This opcode provides a mechanism to export customer data to an external or legacy system for processing when new services have been added to existing customers.

This opcode is called by the PCM_OP_CUST_MODIFY_CUSTOMER standard opcode.

See the discussion on creating hooks to external programs in *BRM Managing Customers*.

PCM_OP_CUST_POL_PRE_COMMIT

Registration hook after account creation. This operation provides a mechanism to easily insert a trigger to external or legacy systems during customer registration.

The PCM_OP_CUST_COMMIT_CUSTOMER opcode calls this opcode just after the **/account** and **/service** objects have been created and initialized, but before the transaction containing those operations has been committed. This opcode cannot alter the contents of the **/account** and **/service** objects, but it can abort the registration process by returning an **ebuf** error.

See the discussion on creating hooks to external programs in *BRM Managing Customers*.

PCM_OP_CUST_POL_PRE_DELETE_PAYINFO

Used to perform custom actions on a **/payinfo** object prior to the deletion of that **/payinfo** object.

This policy opcode is called by the PCM_OP_CUST_DELETE_PAYINFO opcode. The PCM_OP_CUST_DELETE_PAYINFO opcode calls this opcode before deleting a **/payinfo** object.

This policy opcode requires the POID of a **/payinfo** object in the PIN_FLD_POID field as input. It returns the POID of the **/payinfo** object in the PIN_FLD_POID field as its output.

Use this policy opcode to perform any actions that may be necessary before a **/payinfo** object is deleted by the PCM_OP_CUST_DELETE_PAYINFO opcode. 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 opcode.

See the discussion on customizing customer payment information in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_AACINFO

Prepares automatic account creation (AAC) data for validation. This operation takes the AAC fields for an **/account** and **/service** object during customer registration, and processes them as necessary to prepare for validation. This opcode can be used to prepare ACC info to be ready for on-line registration.

The default implementation does nothing.

This opcode is called by the PCM_OP_CUST_INIT_SERVICE and PCM_OP_CUST_ACCTINFO standard opcodes.

See the discussion on customizing automatic account creation (AAC) information in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_ACCTINFO

Prepares account data for validation. This opcode is called before an account is created or modified. This opcode sends the data to PCM_OP_CUST_POL_VALID_ACCTINFO for validation.

This opcode prepares the account information only if the PCM_OP_FLAG_CUST_REGISTRATION flag is set.

This opcode is called by the PCM_OP_CUST_ACCTINFO standard opcode.

See the discussion on the PREP and VALID opcodes in *XBRM Developer's Guide*.

Example 1–130 Sample Input Flist

```
0 PIN_FLD_ACCTINFO      ARRAY [0] allocated 20, used 14
1     PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 / 0 0
1     PIN_FLD_CURRENCY        INT [0] 840
1     PIN_FLD_BUSINESS_TYPE   ENUM [0] 1
1     PIN_FLD_AAC_ACCESS      STR [0] NULL str ptr
1     PIN_FLD_AAC_SOURCE      STR [0] NULL str ptr
1     PIN_FLD_AAC_VENDOR      STR [0] NULL str ptr
1     PIN_FLD_AAC_PACKAGE     STR [0] NULL str ptr
1     PIN_FLD_AAC_PROMO_CODE    STR [0] NULL str ptr
1     PIN_FLD_AAC_SERIAL_NUM    STR [0] NULL str ptr
1     PIN_FLD_ACCOUNT_NO      STR [0] "22825:1:shark"
```

Example 1–131 Sample Output Flist

```
0 PIN_FLD_ACCTINFO      ARRAY [0] allocated 20, used 17
1     PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 / 0 0
1     PIN_FLD_CURRENCY        INT [0] 840
1     PIN_FLD_BUSINESS_TYPE   ENUM [0] 1
1     PIN_FLD_AAC_ACCESS      STR [0] NULL str ptr
1     PIN_FLD_AAC_SOURCE      STR [0] NULL str ptr
1     PIN_FLD_AAC_VENDOR      STR [0] NULL str ptr
1     PIN_FLD_AAC_PACKAGE     STR [0] NULL str ptr
1     PIN_FLD_AAC_PROMO_CODE    STR [0] NULL str ptr
1     PIN_FLD_AAC_SERIAL_NUM    STR [0] NULL str ptr
1     PIN_FLD_ACCOUNT_NO      STR [0] "22825:1:shark"
1     PIN_FLD_GL_SEGMENT      STR [0] "."
1     PIN_FLD_CURRENCY_SECONDARY    INT [0] 0
```

PCM_OP_CUST_POL_PREP_BILLINFO

Prepares billing data for validation. This opcode processes the account billing fields in the **/billinfo** object during customer registration or while updating billing information to prepare for validation.

This opcode is called by the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and PCM_OP_CUST_SET_BILLINFO standard opcodes.

See the discussion on preparing **/billinfo** data in *BRM Configuring and Running Billing*.

PCM_OP_CUST_POL_PREP_INHERITED

Prepares inherited information when a service object is created.

This opcode is called by PCM_OP_CUST_INIT_SERVICE to prepare inherited information to be ready for online registration. This opcode creates a place holder 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.

See the discussion on creating customization interfaces in *BRM Developer's Guide*.

PCM_OP_CUST_POL_PREP_LIMIT

Prepares credit limit information prior to validation.

This opcode is called by the PCM_OP_BILL_SET_LIMIT_AND_CR standard opcode.

- For information about setting credit limits, see the discussion on changing a customer's credit limit in *BRM Managing Customers*.
- For information about the PREP opcodes, see the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

The default implementation does nothing.

PCM_OP_CUST_POL_PREP_LOCALE

Prepares locale information prior to validation.

The default implementation does nothing.

This opcode is called by the PCM_OP_CUST_SET_LOCALE standard opcode.

See the discussion on managing and customizing locale information in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_LOGIN

Prepares service login data for validation. This operation takes the login field for a service object during customer registration and processes it as necessary to prepare for validation.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER and PCM_OP_CUST_SET_LOGIN standard opcodes.

See the discussion on customizing login names in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_NAMEINFO

Prepares customer contact data for validation. This operation takes an element of contact information for an **/account** object during customer registration and processes it as necessary to prepare for validation.

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.

This opcode is called by the PCM_OP_CUST_SET_NAMEINFO and PCM_OP_CUST_COMMIT_CUSTOMER standard opcodes.

See the discussion on customizing name and address information in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_PASSWD

Prepares account or service password for validation. This operation takes the password field for an **/account** or **/service** object during customer registration and processes it as necessary to prepare for validation.

This opcode is called by the PCM_OP_CUST_SET_PASSWD standard opcode.

See the discussion on creating passwords in *BRM Managing Customers*.

PCM_OP_CUST_POL_PREP_PAYINFO

Processes inherited fields and prepares a **/payinfo** object. This opcode checks the pay type and creates the correct object based on that information.

This opcode is called by the PCM_OP_CUST_VALIDATE_CUSTOMER standard opcode.

See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

PCM_OP_CUST_POL_PREP_PROFILE

Modifies account data prior to issuing the final call. Normalizes data for searching purposes.

This opcode is called by the PCM_OP_CUST_CREATE_PROFILE and PCM_OP_CUST_MODIFY_PROFILE standard opcodes, and returns the flist that comes in.

See the discussions on the PREP and VALID opcodes in *BRM Developer's Guide* and collecting nonstandard account information in *BRM Managing Customers*.

Default implementation

The default implementation does nothing.

PCM_OP_CUST_POL_PREP_STATUS

Prepares status information of an account or service prior to validation. This call is used to modify status information before changing the account.

The default implementation does nothing.

This opcode is called by the PCM_OP_CUST_SET_STATUS standard opcode.

See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

PCM_OP_CUST_POL_PREP_TOPUP

Prepares information used to set up and modify standard top-ups and sponsored top-ups.

This opcode, which is in the **fm_cust_pol_prep_topup.c** file, can be customized in many ways to change the way top-ups are set up and modified. For example, you can customize the opcode to enable member accounts to change their top-up PINs and membership status.

This opcode is not called by any opcode.

See the following discussions in *BRM Configuring and Collecting Payments*:

- Preparing an account's top-up information
- Setting sponsored top-up member PINs

PCM_OP_CUST_POL_READ_PLAN

Used during account creation to customize a deal. For a given plan, this opcode constructs a tree of services, deals and products associated with that plan. This opcode retrieves account-level plans in addition to plans related to services.

This opcode is not called by any opcode.

See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.

PCM_OP_CUST_POL_SET_BRANDINFO

Allows brand names to be changed.

To prevent users from creating duplicate brand names within a brand and its sub-brands (the default) or within a BRM system, set the **check_unique** flag. To allow duplicate brand names, disable this check.

This opcode is called by the PCM_OP_CUST_SET_BRANDINFO standard opcode.

See the discussion on changing the brand of an account by using a custom application in *BRM Managing Customers*.

PCM_OP_CUST_POL_TAX_CALC

Queries the tax data loaded in the cache and uses that data to calculate taxes.

By default, this opcode reads the custom tax rate cached from the tax codes map file and uses this simple calculation method to calculate the taxes:

```
tax = amount * rate
```

This opcode is called by the PCM_OP_RATE_TAX_CALC standard opcode.

See the discussion on using custom tax rates in *BRM Configuring and Running Billing*.

PCM_OP_CUST_POL_TAX_INIT

Loads and caches any tax data that you define for calculating taxes. If you use your own tax calculation method instead of using tax calculation software, you use this opcode to load and cache your custom tax rates when the Connection Manager (CM) starts. Then, you can use your custom rates to calculate taxes.

This opcode is not called by any other opcode.

See the discussion on using custom tax rates in *BRM Configuring and Running Billing*.

PCM_OP_CUST_POL_TRANSITION_DEALS

Returns the list of deals and products available for transition. Use this policy opcode to perform any additional filtering of deals before they are returned as available for transition. For example, use this opcode to limit certain deals to customers in a specific city.

This opcode is not called by any opcode.

See the discussion on customizing deal transitions in *BRM Managing Customers*.

Example 1–132 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /deal 11822 0
0 PIN_FLD_TRANSITION_TYPE       ENUM [0] 1
```

Example 1–133 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /deal 8275 0
0 PIN_FLD_DEALS         ARRAY [0] allocated 13, used 13
1     PIN_FLD_POID           POID [0] 0.0.0.1 /deal 10323 0
1     PIN_FLD_CREATED_T    TSTAMP [0] (1085078839) Thu May 20 11:47:19 2004
1     PIN_FLD_MOD_T        TSTAMP [0] (1085078839) Thu May 20 11:47:19 2004
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_DESCR           STR [0] "dealB"
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "dealB"
1     PIN_FLD_PERMITTED       STR [0] "/service/email"
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_PRODUCTS      ARRAY [0] allocated 21, used 21
2         PIN_FLD_CYCLE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 9619 0
2         PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
2         PIN_FLD_USAGE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_USAGE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_USAGE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_NAME            STR [0] "ProductB"
2         PIN_FLD_DESCR           STR [0] "ProductB"
0 PIN_FLD_DEALS         ARRAY [1] allocated 13, used 13
1     PIN_FLD_POID           POID [0] 0.0.0.1 /deal 8814 0
1     PIN_FLD_CREATED_T    TSTAMP [0] (1083899705) Thu May 06 20:15:05 2004
1     PIN_FLD_MOD_T        TSTAMP [0] (1083899705) Thu May 06 20:15:05 2004
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
```

```
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_DESCR           STR [0] ""
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "Deal 1a - Measured Internet Service"
1     PIN_FLD_PERMITTED       STR [0] "/service/ip"
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_PRODUCTS      ARRAY [0] allocated 21, used 21
2         PIN_FLD_CYCLE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 11054 0
2         PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
2         PIN_FLD_USAGE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_USAGE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_USAGE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_NAME            STR [0] "Product 1a - Internet Access"
2         PIN_FLD_DESCR           STR [0] "Charges for monthly internet access
service and hourly usage."
0 PIN_FLD_DEALS         ARRAY [2] allocated 13, used 13
1     PIN_FLD_POID           POID [0] 0.0.0.1 /deal 10862 0
1     PIN_FLD_CREATED_T    TSTAMP [0] (1083899705) Thu May 06 20:15:05 2004
1     PIN_FLD_MOD_T        TSTAMP [0] (1083899705) Thu May 06 20:15:05 2004
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_DESCR           STR [0] ""
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "Deal 1b - Standard Email Access"
1     PIN_FLD_PERMITTED       STR [0] "/service/email"
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_PRODUCTS      ARRAY [0] allocated 21, used 21
2         PIN_FLD_CYCLE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 8878 0
2         PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
2         PIN_FLD_USAGE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_USAGE_END_CYCLE DECIMAL [0] 0
```

```
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_CYCLE DECIMAL [0] 0
2         PIN_FLD_USAGE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_NAME            STR [0] "Product 1b - Email Account"
2         PIN_FLD_DESCR           STR [0] "Charges monthly for 1 email account."
```

PCM_OP_CUST_POL_TRANSITION_PLANS

Returns the list of plans available for transition. Use this policy opcode to perform any additional filtering of plans before they are returned as available for transition. For example, you can use this opcode to limit certain plans to customers in a specific city.

This opcode is not called by any opcode.

See the discussion on customizing deal transitions in *BRM Managing Customers*.

Example 1–134 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /plan 15044 0
0 PIN_FLD_TRANSITION_TYPE  ENUM [0] 0
```

Example 1–135 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /plan 15044 0
0 PIN_FLD_PLAN          ARRAY [0] allocated 11, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /plan 15180 0
1     PIN_FLD_CREATED_T    TSTAMP [0] (1036008525) Wed Oct 30 12:08:45 2002
1     PIN_FLD_MOD_T        TSTAMP [0] (1036008525) Wed Oct 30 12:08:45 2002
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 / 0 0
1     PIN_FLD_DESCR           STR [0] ""
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "Plan_ODB008"
1     PIN_FLD_SERVICES      ARRAY [0] allocated 3, used 3
2         PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 15948 0
2         PIN_FLD_SERVICE_ID      STR [0] ""
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip -1 0
0 PIN_FLD_PLAN              ARRAY [1] allocated 12, used 12
1     PIN_FLD_POID           POID [0] 0.0.0.1 /plan 21582 0
1     PIN_FLD_CREATED_T    TSTAMP [0] (1036139153) Fri Nov 01 00:25:53 2002
1     PIN_FLD_MOD_T        TSTAMP [0] (1036139153) Fri Nov 01 00:25:53 2002
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 / 0 0
1     PIN_FLD_DESCR           STR [0] "Purchase On Demand 50$"
1     PIN_FLD_FLAGS           INT [0] 1048576
1     PIN_FLD_NAME            STR [0] "Purchase On Demand 50$"
1     PIN_FLD_LIMIT         ARRAY [840] allocated 3, used 3
2         PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
2         PIN_FLD_CREDIT_LIMIT DECIMAL [0] 1000
2         PIN_FLD_CREDIT_THRESHOLDS  INT [0] 0
1     PIN_FLD_SERVICES      ARRAY [0] allocated 3, used 3
2         PIN_FLD_DEAL_OBJ   POID [0] 0.0.0.1 /deal 22606 0
2         PIN_FLD_SERVICE_ID  STR [0] ""
2         PIN_FLD_SERVICE_OBJ POID [0] 0.0.0.1 /service/ip -1 0
```

PCM_OP_CUST_POL_VALID_AACINFO

Validates automatic account creation data. This operation takes the automatic account creation fields for an **/account** or **/service** object during customer registration, and validates them.

This opcode is called by the PCM_OP_CUST_INIT_SERVICE standard opcode.

See the discussion on customizing automatic account creation (AAC) information in *BRM Managing Customers*.

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 the PCM_OP_CUST_POL_PREP_ACCTINFO opcode when an account is being created or modified. If the mandatory fields are not passed in, this opcode reports an error.

By default, this policy opcode does not modify anything. You need to modify this opcode only for special situations.

This opcode is called by the PCM_OP_CUST_VALIDATE_CUSTOMER standard opcode.

See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

Example 1–136 Sample Input Flist

```
0 PIN_FLD_ACCTINFO      ARRAY [0] allocated 20, used 17
1     PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0 / 0 0
1     PIN_FLD_BAL_INFO      ARRAY [0]     NULL array ptr
1     PIN_FLD_CURRENCY        INT [0] 840
1     PIN_FLD_BUSINESS_TYPE   ENUM [0] 1
1     PIN_FLD_AAC_ACCESS      STR [0] NULL str ptr
1     PIN_FLD_AAC_SOURCE      STR [0] NULL str ptr
1     PIN_FLD_AAC_VENDOR      STR [0] NULL str ptr
1     PIN_FLD_AAC_PACKAGE     STR [0] NULL str ptr
1     PIN_FLD_AAC_PROMO_CODE    STR [0] NULL str ptr
1     PIN_FLD_AAC_SERIAL_NUM    STR [0] NULL str ptr
1     PIN_FLD_NAME            STR [0] "PIN Account Object"
1     PIN_FLD_ACCOUNT_TYPE   ENUM [0] 0
1     PIN_FLD_ACCOUNT_NO      STR [0] "22825:1:shark"
1     PIN_FLD_ACTG_TYPE      ENUM [0] 2
1     PIN_FLD_GL_SEGMENT      STR [0] "."
1     PIN_FLD_CURRENCY_SECONDARY    INT [0] 0
```

Example 1–137 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 0
0 PIN_FLD_RESULT         ENUM [0] 1
```

PCM_OP_CUST_POL_VALID_BILLINFO

Validates billing information. This opcode validates an account's billing information in the **/billinfo** object passed to it by PCM_OP_CUST_POL_PREP_BILLINFO during customer registration or administrative update.

This opcode is called by the PCM_OP_CUST_VALIDATE_CUSTOMER and PCM_OP_CUST_SET_BILLINFO standard opcodes.

See the discussion on validating billinfo data in *BRM Configuring and Running Billing*.

PCM_OP_CUST_POL_VALID_LIMIT

Validates credit limit information before it is set in the account.

This opcode is called by the PCM_OP_BILL_SET_LIMIT_AND_CR standard opcode.

See the discussion on how BRM handles consumption rules and credit limits in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_LOCALE

Validates locale information before it is set in the account.

This opcode is called by the PCM_OP_CUST_SET_LOCALE and PCM_OP_CUST_VALIDATE_CUSTOMER standard opcodes.

See the discussion on managing and customizing locale information in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_LOGIN

Validates the given login according to the criteria contained in the **/config/fld_validate** object. This operation takes the login field for a **/service** object during customer registration or administrative update and validates it.

> **Caution:** BRM requires unique login names for service types. BRM will not function properly if this opcode is customized to allow nonunique login names.

This opcode is called by the PCM_OP_CUST_SET_LOGIN, PCM_OP_CUST_COMMIT_CUSTOMER, and PCM_OP_CUST_VALIDATE_CUSTOMER standard opcodes.

See the discussion on customizing login names in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_NAMEINFO

Validates customer contact data. This operation takes an element of contact information for an **/account** object during customer registration or administrative update, and validates the fields in it.

This opcode is called by the PCM_OP_CUST_SET_NAMEINFO and PCM_OP_CUST_VALIDATE_CUSTOMER standard opcodes.

See the discussion on customizing name and address information in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_PASSWD

Validates account or service password data. This operation takes the password field for an **/account** or **/service** object during customer registration or administrative update and validates it.

The default check is to make sure that the password is not NULL and is less than 255 characters.

This opcode is called by the PCM_OP_CUST_SET_PASSWD and PCM_OP_CUST_VALIDATE_CUSTOMER standard opcodes.

See the discussion on creating passwords in *BRM Managing Customers*.

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 transactions. For credit cards, this opcode checks the credit card type, number, expiration date, and CVV2 or CID number during registration.

> **Note:** The CVV2 and CID numbers are used by Visa and American Express for credit card fraud prevention. If the CM **pin.conf** file's **cvv2_required** flag is set to **1** (required) Andes 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*".

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

This opcode is called by the PCM_OP_CUST_VALIDATE_CUSTOMER standard opcode.

See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

PCM_OP_CUST_POL_VALID_PROFILE

Reviews and validates data prior to creating an object.

This opcode is called by the PCM_OP_CUST_CREATE_PROFILE, PCM_OP_CUST_VALIDATE_CUSTOMER, and PCM_OP_CUST_MODIFY_PROFILE standard opcodes, and returns the flist that comes in. If the data is not valid, a list of possible problems is returned.

The default implementation does nothing.

See the discussion on collecting nonstandard account information in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_STATUS

Validates status information before it is set in the account. This operation validates status information for an account or service.

The default is to do no additional checking and to return the verified information.

This opcode is called by the PCM_OP_CUST_SET_STATUS standard opcode.

See the discussion on changing the status of an account, bill unit, or service in *BRM Managing Customers*.

PCM_OP_CUST_POL_VALID_TAXINFO

Validates the VAT certificate number provided during account creation.

During account creation, the PCM_OP_CUST_SET_TAXINFO standard opcode calls this opcode to validate the VAT certificate number provided. This opcode prevents invalid VAT certificate numbers which cause errors in tax calculation.

This opcode returns the validation results of **PASS** or **FAIL**.

See the discussion on validating tax information in *BRM Configuring and Running Billing*.

PCM_OP_CUST_POL_VALID_TOPUP

Validates information used to set up and modify standard top-ups and sponsored top-ups. You can customize this opcode to change the way it validates the output flist of the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.

This opcode is not called by any opcode.

See the discussion on validating an account's top-up information in *BRM Configuring and Collecting Payments*.

Customer FM Standard Opcodes

The opcodes in Table 1–24 manage the creation, deletion, and modification of account information during customer registration.

Header File

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

Opcode Index

Table 1–24 Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_AMEND_CREDITOR_INFO	Updates the creditor details in the **/config/creditor** object and applies the creditor update to the **/payinfo/sepa** objects that are impacted by the amendment of the creditor information. See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_CUST_AMEND_MANDATE	Updates the mandate fields, except for the creditor information, in the **/payinfo/sepa** object. See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_CUST_CANCEL_MANDATE	Cancels a mandate. See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_CUST_CHANGE_BUSINESS_PROFILE	Changes a bill unit's business profile. Validates all the balance groups and services associated with the bill unit against the requirements of the new business profile. See the discussion on changing a bill unit's business profile in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_COMMIT_CUSTOMER	Performs all the tasks necessary to create an active and billable account in the database. See the discussion on how BRM creates accounts in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_CREATE_ACCT	Creates a generic **/account** object. See the discussion on how BRM creates accounts in *BRM Managing Customers*.	Last Resort

Table 1–24 (Cont.) Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE	Creates one **/associated_bus_profile** object for each bill unit in the account. See the discussion on associating bill units with a BI Publisher invoice and report in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_CREATE_BAL_GRP	Creates an active **/balance_group** object. See the discussion on creating balance groups in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_CUST_CREATE_BILLINFO	Creates an active **/billinfo** object. See the discussion on creating **/billinfo** objects in *BRM Configuring and Running Billing*.	Limited
PCM_OP_CUST_CREATE_CUSTOMER	Creates an active customer account, including creating and initializing an **/account** object and one or more **/service** objects. See the discussion on how BRM creates accounts in *BRM Managing Customers*.	Limited
PCM_OP_CUST_CREATE_PAYINFO	Creates an active **/payinfo** object. See the discussions on how BRM creates accounts and on customizing customer payment information in *BRM Managing Customers*.	Limited
PCM_OP_CUST_CREATE_PROFILE	Creates a **/profile** object. See the discussion on managing and customizing profiles in *BRM Managing Customers*.	Limited
PCM_OP_CUST_CREATE_SERVICE	Creates an active service object with inheritance pass through. See the discussion on creating services in *BRM Managing Customers*.	Last Resort
PCM_OP_CUST_CREATE_TOPUP	Creates **/topup** and **/group/topup** objects. See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_CUST_DELETE_ACCT	Deletes the specified **/account** object and all related objects. See the discussion on deleting accounts in *BRM Managing Customers*.	Last Resort
PCM_OP_CUST_DELETE_BAL_GRP	Deletes the specified **/balance_group** object. See the discussion on deleting a balance group in *BRM Managing Accounts Receivable*.	Last Resort

Table 1–24 (Cont.) Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_DELETE_BILLINFO	Deletes the specified **/billinfo** object and the balance groups associated with it. See the discussion on deleting billinfo objects in *BRM Configuring and Running Billing*.	Last Resort
PCM_OP_CUST_DELETE_PAYINFO	Deletes the specified **/payinfo** object. See the discussion on customizing customer payment information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_DELETE_PROFILE	Deletes the specified **/profile** object. See the discussion on managing and customizing profiles in *BRM Managing Customers*.	Limited
PCM_OP_CUST_DELETE_TOPUP	Deletes **/topup** objects. See the discussion on deleting member accounts in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_CUST_FIND	Searches for information in an **/account** object given an account number. See the discussion on finding customer accounts using opcodes in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_FIND_PAYINFO	Finds **/payinfo** objects that belong to a specified account. See the discussion on finding payment info in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_CUST_FIND_PROFILE	Retrieves the **/profile** objects associated with a specified account. See the discussion on searching for account profile information in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO	Gets the value of a key in the business profile and validation templates associated with an object. See the discussion on getting information about an object's business profile in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_GET_LIFECYCLE_STATES	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 the discussion on managing service life cycles in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_GET_NEWSFEED	Retrieves and returns the News Feed information stored in the **/newsfeed** object.	Recommended
PCM_OP_CUST_GET_NOTE	Gets notes and note exchanges based on any of the **/note** object fields.	Recommended

Table 1–24 (Cont.) Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES	Retrieves specific subscriber preferences (if indicated in the input list) or all the preferences for a subscriber. See the discussions on how the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode works and on maintaining a subscriber's preferences data with custom client applications in *BRM Telco Integration*.	Recommended
PCM_OP_CUST_INIT_SERVICE	Initializes a service object. See the discussion on creating services in *BRM Managing Customers*.	Last Resort
PCM_OP_CUST_MODIFY_BAL_GRP	Modifies the specified **/balance_group** object. See the discussion on managing balance groups with your custom application in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_CUST_MODIFY_CUSTOMER	Modifies customer account information. See the discussion on modifying an account in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_MODIFY_PAYINFO	Modifies selected fields in the **/payinfo** object. See the discussion on customizing customer payment information in *BRM Managing Customers*.	Limited
PCM_OP_CUST_MODIFY_PROFILE	Modifies the specified **/profile** object. See the discussion on managing and customizing profiles in *BRM Managing Customers*.	Limited
PCM_OP_CUST_MODIFY_SERVICE	Modifies the specified **/service** object. See the discussion on modifying services in *BRM Managing Customers*.	Limited
PCM_OP_CUST_MODIFY_TOPUP	Modifies **/topup** and **/group/topup** objects. See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_CUST_PREP_CUSTOMER	Validates customer registration information prior to account creation. See the discussion on how BRM creates accounts in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_SET_ACCTINFO	Initializes an **/account** object with generic fields passed in on the input flist. See the discussion on how BRM creates accounts in *BRM Managing Customers*.	Limited

Table 1–24 (Cont.) Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE	Updates the **/associated_bus_profile** objects whenever invoice business profiles are modified in the **/config/business_profile** object. See the discussion on associating bill units with a BI Publisher invoice and report in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_SET_BAL_GRP	Performs all necessary tasks to set up the **/balance_group** object and create a link to the customer account. See the discussion on managing balance groups with your custom application in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_CUST_SET_BILLINFO	Updates billing information in a bill unit (**/billinfo** object) for a specified account. See the discussion on creating billinfo objects in *BRM Configuring and Running Billing*.	Limited
PCM_OP_CUST_SET_BRANDINFO	Enables changing the brand name after account creation. See the discussion on managing brands in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_SET_LOCALE	Sets the Locale field of a specified **/account** object. See the discussion on managing and customizing locale information in *BRM Managing Customers*.	Limited
PCM_OP_CUST_SET_LOGIN	Updates the service login. See the discussion on customizing login names in *BRM Managing Customers*.	Limited
PCM_OP_CUST_SET_NAMEINFO	Sets account contact information such as customer name, address, and phone number. See the discussion on managing customer contact information in *BRM Managing Customers*.	Limited
PCM_OP_CUST_SET_NOTE	Creates a new **/note** object or modifies an existing one.	Recommended
PCM_OP_CUST_SET_PASSWD	Updates the service password for a customer. See the discussion on customizing passwords in *BRM Managing Customers*.	Limited
PCM_OP_CUST_SET_PAYINFO	Adds or updates the payment information for a bill unit (**/billinfo** object). See the discussion on customizing customer payment information in *BRM Managing Customers*.	Limited

Table 1–24 (Cont.) Customer FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUST_SET_STATUS	Sets the status of an **/account**, **/billinfo**, or **/service** object. See the discussion on setting account, service, and bill unit status by using your custom application in *BRM Managing Customers*.	Limited
PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES	Creates, modifies, or deletes a **/profile/subscriber_preferences** object for a specified service or account. See the discussions on how the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode works and on maintaining a subscriber's preferences data with custom client applications in *BRM Telco Integration*.	Recommended
PCM_OP_CUST_SET_TAXINFO	Adds or updates tax-related fields of an account. See the discussion on adding tax information to accounts in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_CUST_SET_TOPUP	Sets up standard top-ups and sponsored top-ups. See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_CUST_UPDATE_CUSTOMER	Updates several pieces of customer information in one operation. See the discussion on modifying an account in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_UPDATE_SERVICES	Modifies service information for multiple services in one operation. See the discussion on creating services in *BRM Managing Customers*.	Recommended
PCM_OP_CUST_VALID_FLD	Validates fields on the input flist based on the information contained in the **/config/fld_validate** object. See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.	Recommended
PCM_OP_CUST_VALIDATE_CUSTOMER	Validates customer information during registration. See the discussion on validating data from account creation applications in *BRM Managing Customers*.	Recommended

PCM_OP_CUST_AMEND_CREDITOR_INFO

Updates the creditor name and creditor identification in the **/config/creditor** object and applies the creditor update to the **/payinfo/sepa** objects that are impacted by the amendment of the creditor information.

See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_AMEND_MANDATE

Updates the mandate fields, except for the creditor information, in the **/payinfo/sepa** object.

See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_CANCEL_MANDATE

Cancels a mandate in the **/payinfo/sepa** object by setting the mandate status to canceled.

See the discussion about SEPA payment processing in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_CHANGE_BUSINESS_PROFILE

Changes a bill unit's business profile. Validates all the balance groups and services associated with the bill unit against the requirements of the new business profile.

See the discussion on changing a bill unit's business profile in *BRM Managing Customers*.

PCM_OP_CUST_COMMIT_CUSTOMER

A wrapper opcode that performs all the tasks necessary to create an active and billable account in the database.

See the discussion on how BRM creates accounts in *BRM Managing Customers*.

Note: 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*.

Transaction Handling

The value of the input flist PIN_FLD_TXN_FLAGS field determines how this opcode handles transactions:

- When set to **1**, this opcode *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 this opcode is not called by JCA Resource Adapter.

  Note: After the credit card authorization, if errors occur in the account creation processes, an account might be partially created.

- When set to **2**, this opcode *does not open its own transaction*. Instead, it uses the transaction already opened by the application server in XA or local transaction mode.

  When JCA Resource Adapter is in XA or local transaction mode and receives a request to call this opcode, 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 this opcode to support transactions opened by a global transaction manager, the PIN_FLD_TXN_FLAGS field must be set to **2**.

This opcode returns an error in the following situations:

- A transaction is already open when this opcode 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 this opcode in XA or local transaction mode. For information about the adapter's transaction modes, see

"About JCA Resource Adapter Transaction Management" in *BRM JCA Resource Adapter*.

PCM_OP_CUST_CREATE_ACCT

Creates a generic **/account** object.

See the discussion on how BRM creates accounts in *BRM Managing Customers*.

PCM_OP_CUST_CREATE_ASSOCIATED_BUS_PROFILE

Creates one **/associated_bus_profile** object for each bill unit in the account.

If the BRM-BI Publisher invoice integration is enabled, during customer account creation, internally the PCM_OP_CUST_CREATE_BILLINFO opcode calls this opcode to create one **/associated_bus_profile** object for each bill unit in the account.

See the discussion on associating bill units with a BI Publisher invoice and report in *BRM Configuring and Running Billing*.

PCM_OP_CUST_CREATE_BAL_GRP

Creates an active **/balance_group** object.

See the discussion on creating balance groups in *BRM Managing Accounts Receivable*.

This opcode is called during account creation. The wrapper opcode PCM_OP_CUST_SET_BAL_GRP calls this opcode to create the **/balance_group** object.

Example 1–138 Sample Input Flist

```
0 PIN_FLD_POID   POID [0] 0.0.0.1 /balance_group -1 0
0 PIN_FLD_BILLINFO_OBJ POID [0] 0.0.0.1 /billinfo 199680 0
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 197632 0
```

Example 1–139 Sample Output Flist

```
0 PIN_FLD_POID   POID [0] 0.0.0.1 /balance_group 200192 0
```

PCM_OP_CUST_CREATE_BILLINFO

Creates an active **/billinfo** object.

See the discussion on creating **/billinfo** objects in *BRM Configuring and Running Billing*.

Example 1–140 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 53
0 PIN_FLD_PROGRAM_NAME    STR [0] "test"
0 PIN_FLD_BILLINFO      ARRAY [1]
1 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo -1
1 PIN_FLD_PAY_TYPE      ENUM [0] 10001
1 PIN_FLD_PAYINFO_OBJ           POID [0] 0.0.0.1 /payinfo/invoice 555555
1 PIN_FLD_BILLINFO_ID    STR [0] "my_billinfo"
1 PIN_FLD_CURRENCY        INT [0] 840
1 PIN_FLD_CURRENCY_SECONDARY        INT [0] 0
```

Example 1–141 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 53
0 PIN_FLD_BILLINFO      ARRAY [1]
1 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 1363
1 PIN_FLD_BILLINFO_ID    STR [0] "my_billinfo"
```

PCM_OP_CUST_CREATE_CUSTOMER

Creates an active customer account, including creating and initializing an **/account** object and one or more **/service** objects.

No validation is performed by this opcode 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 input flist for this opcode should be taken from the output of a call to PCM_OP_CUST_PREP_CUSTOMER to ensure the fields have been properly validated.

Important: If you use rerating, use the PCM_OP_CUST_COMMIT_CUSTOMER opcode to create accounts. Do not call the PCM_OP_CUST_CREATE_CUSTOMER opcode directly. The PCM_OP_CUST_COMMIT_CUSTOMER opcode calls the PCM_OP_CUST_POL_PRE_COMMIT policy opcode, and based on that, the **/profile/event_ordering** object is created, which is used for rerating. If you use the PCM_OP_CUST_CREATE_CUSTOMER opcode directly to create accounts, the **/profile/event_ordering** object is not created.

If balance monitoring is enabled, this opcode passes balance monitor data to PCM_OP_CUST_SET_BAL_GRP.

See the following discussions:

- How BRM creates accounts in *BRM Managing Customers*
- Balance monitoring in *BRM Managing Accounts Receivable*

PCM_OP_CUST_CREATE_PAYINFO

Creates an active **/payinfo** object.

This opcode also updates the PIN_FLD_PAYINFO_OBJ field in the **/billinfo** object.

This opcode is called during customer registration.

> **Note:** For credit card payment methods, this opcode omits the PIN_FLD_SECURITY_ID field from the input flist of PCM_OP_CREATE_OBJ when the **/payinfo/cc** object is created. The result is that the CVV2/CID information is stored in the database with a NULL value.

See the discussions on how BRM creates accounts and on customizing customer payment information in *BRM Managing Customers*.

PCM_OP_CUST_CREATE_PROFILE

Creates a **/profile** object.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated ERA modifications when certain conditions are met.

> **Important:** Only one element can be passed in the PIN_FLD_PROFILES array. Otherwise, the opcode ignores the array.

See the following discussions:

- Managing and customizing profiles in *BRM Managing Customers*
- Backdated ERA modifications in *BRM Configuring and Running Billing*

PCM_OP_CUST_CREATE_SERVICE

Creates an active service object with inheritance pass through.

See the discussion on creating services in *BRM Managing Customers*.

PCM_OP_CUST_CREATE_TOPUP

Creates **/topup** and **/group/topup** objects.

This opcode is called by the wrapper opcode PCM_OP_CUST_SET_TOPUP.

See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_DELETE_ACCT

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.

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.

See the discussion on deleting accounts in *BRM Managing Customers*.

Caution: Do not delete accounts in a production system.

Note: 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 an account receivable (A/R) bill unit of another existing account. For information about subscription service transfer, see the discussion on transferring a subscription service to another subscriber in *BRM Managing Customers*. For information about A/R and hierarchical account groups, see the discussion on A/R and hierarchical account groups in *BRM Managing Accounts Receivable*.

PCM_OP_CUST_DELETE_BAL_GRP

Deletes the specified **/balance_group** object.

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.

If successful, the output flist contains the POID of the **/balance_group** object that is deleted.

See the discussion on deleting a balance group in *BRM Managing Accounts Receivable*.

Example 1–142 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 10942
```

Example 1–143 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 10942
```

PCM_OP_CUST_DELETE_BILLINFO

Deletes the specified **/billinfo** object and the balance groups associated with it.

See the discussion on deleting billinfo objects in *BRM Configuring and Running Billing*.

Example 1–144 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 7777
0 PIN_FLD_PROGRAM_NAME    STR [0] "my delete program"
0 PIN_FLD_BILLINFO      ARRAY [1]
1 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 12418
```

Example 1–145 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 7777
0 PIN_FLD_BILLINFO      ARRAY [1]
1 PIN_FLD_POID           POID [0] 0.0.0.1 /billinfo 12418
```

PCM_OP_CUST_DELETE_PAYINFO

Deletes the specified **/payinfo** object.

This opcode 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 this opcode deletes the **/payinfo** object, it calls the PCM_OP_CUST_POL_PRE_DELETE_PAYINFO policy opcode 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).

See the discussion on customizing customer payment information in *BRM Managing Customers*.

PCM_OP_CUST_DELETE_PROFILE

Deletes the specified **/profile** object.

If the profile object specified in the input flist is a part of a profile sharing group (**/group/sharing/profile** object), this opcode 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.

See the discussion on managing and customizing profiles in *BRM Managing Customers*.

PCM_OP_CUST_DELETE_TOPUP

Deletes **/topup** objects.

This opcode is called by the PCM_OP_CUST_DELETE_ACCT opcode.

See the discussion on deleting member accounts in *BRM Configuring and Collecting Payments*.

Important: This opcode should not be used to cancel an account's membership in a sponsored top-up group. See the discussion on canceling top-ups in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_FIND

Searches for information in an **/account** object given an account number.

See the discussion on finding customer accounts using opcodes in *BRM Managing Customers*.

PCM_OP_CUST_FIND_PAYINFO

Finds **/payinfo** objects that belong to a specified account.

This opcode is given the account POID and returns the information from the storable **/payinfo** object.

See the discussion on finding payment info in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_FIND_PROFILE

Retrieves the **/profile** objects associated with a specified account.

See the discussion on searching for account profile information in *BRM Managing Customers*.

PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO

Gets the value of a key in the business profile and validation templates associated with an object.

See the discussion on getting information about an object's business profile in *BRM Managing Customers*.

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 billinfo POIDs are passed, gets the PIN_FLD_STATES array for the initial state (PIN_FLD_INITIAL_STATE = 1) of the life cycle

This opcode is called by the following components:

- The PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE opcode calls this opcode to get a life cycle states configuration object.
- **pin_state_change** calls this opcode to get state transition information from a life cycle states configuration object.
- The PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE policy opcode calls this opcode to get the state expiration time.
- The PCM_OP_CUST_SET_STATUS opcode calls this opcode to get the service's initial state.
- The PCM_OP_CUST_SET_STATUS and PCM_OP_CUST_UPDATE_SERVICES opcodes call this opcode during validation.
- Customer Center calls this opcode to get state transition information from a life cycle states configuration object so that it can display a list of states to which a service can change from its current state.

This opcode is used only if the **SubscriberLifeCycle** business parameter is associated with the specified service's bill unit and is enabled.

See the discussion on managing service life cycles in *BRM Managing Customers*.

PCM_OP_CUST_GET_NEWSFEED

Retrieves and returns the News Feed information stored in the **/newsfeed** object.

PCM_OP_CUST_GET_NOTE

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

Retrieves the subscriber's preferences from the **/profile/subscriber_preferences** object for a subscriber's service or account.

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.

This opcode is called by the PCM_OP_TCF_AAA_POL_POST_PROCESS opcode. It can also be called by Customer Center or a custom application.

See the discussions on how the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode works and on maintaining a subscriber's preferences data with custom client applications in *BRM Telco Integration*.

PCM_OP_CUST_INIT_SERVICE

Initializes a service object.

This opcode 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.

See the discussion on creating services in *BRM Managing Customers*.

PCM_OP_CUST_MODIFY_BAL_GRP

Modifies the specified **/balance_group** object.

If successful, the output flist contains the POID of the **/balance_group** object that is modified.

See the discussion on managing balance groups with your custom application in *BRM Managing Accounts Receivable*.

Example 1–146 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 198784 0
0 PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 199424 0
```

Example 1–147 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 198784 0
```

PCM_OP_CUST_MODIFY_CUSTOMER

Modifies customer account information.

If balance monitoring is enabled, this opcode passes balance monitor data to PCM_OP_CUST_SET_BAL_GRP.

See the following discussions:

- Modifying an account in *BRM Managing Customers*
- Balance monitoring in *BRM Managing Accounts Receivable*

Note: If the PIN_FLD_STATUS_FLAGS field is set to PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE, this opcode verifies that the service group relationships are valid and associates member services with the appropriate balance group.

PCM_OP_CUST_MODIFY_PAYINFO

Modifies selected fields in the **/payinfo** object.

This opcode is called by PCM_OP_CUST_SET_PAYINFO and calls PCM_OP_WRITE_FLDS. One or more fields must be selected or an error will be returned. The **/payinfo** object is modified only if the data in the input flist is different from the **/payinfo** object data in the database.

Note: This opcode omits the PIN_FLD_SECURITY_ID field from the input flist of PCM_OP_WRITE_FLDS when the **/payinfo/cc** object is updated. The result is that the CVV2/CID information is stored in the database with a NULL value.

See the discussion on customizing customer payment information in *BRM Managing Customers*.

PCM_OP_CUST_MODIFY_PROFILE

Modifies the specified **/profile** object.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated ERA modifications when certain conditions are met.

> **Important:** Only one element can be passed in the PIN_FLD_PROFILES array. Otherwise, the opcode ignores the array.

See the following discussions:

- Managing and customizing profiles in *BRM Managing Customers*
- Backdated ERA modifications in *BRM Configuring and Running Billing*

PCM_OP_CUST_MODIFY_SERVICE

Modifies the specified **/service** object.

For most services, a wrapper opcode calls PCM_OP_CUST_MODIFY_SERVICE to set, change, or delete extended service information.

See the discussion on modifying services in *BRM Managing Customers*.

PCM_OP_CUST_MODIFY_TOPUP

Modifies **/topup** and **/group/topup** objects.

This opcode is called by the wrapper opcode PCM_OP_CUST_SET_TOPUP.

See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_PREP_CUSTOMER

Validates customer registration information prior to account creation.

See the discussion on how BRM creates accounts in *BRM Managing Customers*.

PCM_OP_CUST_SET_ACCTINFO

Initializes an **/account** object with generic fields passed in on the input flist. Calls the PCM_OP_CUST_POL_PREP_ACCTINFO and PCM_OP_CUST_POL_VALID_ACCTINFO opcodes to prepare and validate account information, and returns the account number based on the customizations made in the policy opcodes.

See the discussion on how BRM creates accounts in *BRM Managing Customers*.

PCM_OP_CUST_SET_ASSOCIATED_BUS_PROFILE

Updates the **/associated_bus_profile** objects whenever invoice business profiles are modified in the **/config/business_profile** object.

See the discussion on associating bill units with a BI Publisher invoice and report in *BRM Configuring and Running Billing*.

PCM_OP_CUST_SET_BAL_GRP

A wrapper opcode that performs all necessary tasks to set up the **/balance_group** object and create a link to the customer account.

If balance monitoring is enabled, this opcode creates or updates the **/balance_group/monitor** object.

See the following discussions in *BRM Managing Accounts Receivable*:

- Managing balance groups with your custom application
- Balance monitoring

Example 1–148 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 197632 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "Automatic Account Creation"
0 PIN_FLD_START_T      TSTAMP [0] (1064290628) Mon Sep 22 21:17:08 2003
0 PIN_FLD_END_T        TSTAMP [0] (1064290628) Mon Sep 22 21:17:08 2003
0 PIN_FLD_LOCALE          STR [0] "en_US"
0 PIN_FLD_BAL_INFO      ARRAY [0] allocated 20, used 5
1     PIN_FLD_LIMIT         ARRAY [840] allocated 20, used 1
2         PIN_FLD_CREDIT_LIMIT DECIMAL [0] .0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group -1 0
1     PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 199680 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 197632 0
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.0  0 0
```

Example 1–149 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 200192 0
```

PCM_OP_CUST_SET_BILLINFO

Updates billing information in a bill unit (**/billinfo** object) for a specified account.

This opcode 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.

This opcode calls the PCM_OP_CUST_POL_PREP_BILLINFO policy opcode to prepare the updated billing information for validation and then calls the PCM_OP_CUST_POL_VALID_BILLINFO policy opcode to validate the information.

See the discussion on creating billinfo objects in *BRM Configuring and Running Billing*.

PCM_OP_CUST_SET_BRANDINFO

Enables changing the brand name after account creation.

See the discussion on managing brands in *BRM Managing Customers*.

PCM_OP_CUST_SET_LOCALE

Sets the Locale field of a specified **/account** object.

See the discussion on managing and customizing locale information in *BRM Managing Customers*.

PCM_OP_CUST_SET_LOGIN

Updates the service login for a customer.

See the discussion on customizing login names in *BRM Managing Customers*.

PCM_OP_CUST_SET_NAMEINFO

Sets account contact information such as customer name, address, and phone number.

See the discussion on managing customer contact information in *BRM Managing Customers*.

PCM_OP_CUST_SET_NOTE

Creates a new **/note** object or modifies an existing one. If you pass in a type-only POID, a new **/note** object is created. Otherwise, the opcode modifies the **/note** object.

The opcode generates the following events:

- **/event/customer/note/create**
- **/event/customer/note/modify**

PCM_OP_CUST_SET_PASSWD

Updates the service password for a customer.

See the discussion on customizing passwords in *BRM Managing Customers*.

PCM_OP_CUST_SET_PAYINFO

Adds or updates the payment information for a bill unit (**/billinfo** object).

This opcode is a wrapper for the following opcodes:

- PCM_OP_CUST_CREATE_PAYINFO
- PCM_OP_CUST_MODIFY_PAYINFO
- PCM_OP_CUST_DELETE_PAYINFO

During customer registration, this opcode creates a **/payinfo** object that contains information about how a customer will pay the bill (for example, by credit card, direct debit, invoice, and so on) and attaches the **/payinfo** object to the appropriate bill unit (**/billinfo** object).

During account modification, this opcode modifies the payment information for the bill unit if necessary.

This opcode creates an **/event/audit/customer/payinfo** object.

See the discussion on customizing customer payment information in *BRM Managing Customers*.

PCM_OP_CUST_SET_STATUS

Sets the status of an **/account**, **/billinfo**, or **/service** object.

This opcode triggers auto-billing if bills are still pending.

Note: For service status changes, this opcode is not called directly. The PCM_OP_CUST_UPDATE_SERVICES opcode is called, which in turn calls this opcode.

See the discussion on setting account, service, and bill unit status by using your custom application in *BRM Managing Customers*.

PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES

Creates, modifies, or deletes a **/profile/subscriber_preferences** object for a specified service or account.

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**, this opcode deletes the **/profile/subscriber_preferences** object.

This opcode is called by the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode. It can also be called from Customer Center or a custom application.

See the discussions on how the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode works and on maintaining a subscriber's preferences data with custom client applications in *BRM Telco Integration*.

PCM_OP_CUST_SET_TAXINFO

Adds or updates the tax information in the account object.

This opcode adds the following data:

- VAT certificate
- Tax exemptions
- Tax incorporation
- Tax residence

See the discussion on adding tax information to accounts in *BRM Configuring and Running Billing*.

PCM_OP_CUST_SET_TOPUP

Sets up standard top-ups and sponsored top-ups.

This is a wrapper opcode that calls other standard opcodes to create or modify **/topup** and **/group/topup** objects.

See the discussion on how BRM sets up top-up information for an account in *BRM Configuring and Collecting Payments*.

PCM_OP_CUST_UPDATE_CUSTOMER

Updates customer account information.

See the discussion on modifying an account in *BRM Managing Customers*.

PCM_OP_CUST_UPDATE_SERVICES

Modifies an account's service information for multiple services in one operation.

See the discussion on creating services in *BRM Managing Customers*.

Example 1–150 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /account 12177 0
0 PIN_FLD_PROGRAM_NAME                   STR [0] "testnap"
0 PIN_FLD_FLAGS                          INT [0] 1
0 PIN_FLD_SERVICES                     ARRAY [0] allocated 20, used 4
1      PIN_FLD_POID                     POID [0] 0.0.0.1 /service/email 8785 -1
1      PIN_FLD_LOGIN                     STR [0] "ac1"
1      PIN_FLD_PASSWD_CLEAR              STR [0] "password"
1      PIN_FLD_INHERITED_INFO     SUBSTRUCT [0] allocated 20, used 1
2           PIN_FLD_SERVICE_EMAIL SUBSTRUCT [0] allocated 20, used 1
3                PIN_FLD_PATH            STR [0] "/tmp"
```

Note: PIN_FLD_FLAGS should be used by Telco opcodes only while calling PCM_OP_CUST_UPDATE_SERVICES.

Example 1–151 Sample Output Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /account 12177 0
0 PIN_FLD_SERVICES           ARRAY [0] allocated 4, used 4
1      PIN_FLD_POID           POID [0] 0.0.0.1 /service/email 8785 -1
1      PIN_FLD_RESULTS       ARRAY [0] allocated 2, used 2
2           PIN_FLD_POID      POID [0] 0.0.0.1 /event/customer/login
205810984533632825 0
2           PIN_FLD_LOGINS   ARRAY [1] allocated 1, used 1
3                PIN_FLD_LOGIN STR [0] "ac1@portal.com"
1      PIN_FLD_RESULTS       ARRAY [1] allocated 1, used 1
2           PIN_FLD_POID      POID [0] 0.0.0.1 /event/customer/password
205810984533634873 0
1      PIN_FLD_RESULTS       ARRAY [3] allocated 1, used 1
2      PIN_FLD_POID            POID [0] 0.0.0.1 /service/email 8785 -1
```

PCM_OP_CUST_VALID_FLD

Validates field values on the input flist based on the information contained in the **/config/fld_validate** object.

See the discussion on the PREP and VALID opcodes in *BRM Developer's Guide*.

PCM_OP_CUST_VALIDATE_CUSTOMER

Validates customer information during registration.

During registration, this opcode validates customer information as the information is provided. This opcode can validate partial information. When the user goes to the next screen in an application, information provided on a screen is validated even if the information is not complete.

See the discussion on validating data from account creation applications in *BRM Managing Customers*.

Customer Care FM Standard Opcodes

Use the opcodes listed in Table 1–25 to manage customers.

Header File

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

Opcode Index

Table 1–25 Customer Care FM Standard Opcodes

Opcode	Description	Use
PCM_OP_CUSTCARE_MOVE_ACCT	This opcode moves an account into or out of an account hierarchy.	Recommended

PCM_OP_CUSTCARE_MOVE_ACCT

This opcode moves an account into or out of an account hierarchy.

Device FM Policy Opcodes

Use the opcodes listed in Table 1–26 to customize device management.

Header File

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

Opcode Index

Table 1–26 Device FM Policy Opcodes

Opcode	Description	Use
PCM_OP_DEVICE_POL_ASSOCIATE	Can be customized to provide validation for associations and disassociations. See the discussion on associating service and device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_POL_CREATE	Can be customized to provide validation and other functionality during device creation. See the discussion on creating device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_POL_DELETE	Can be customized to provide validation for device deletions. See the discussion on deleting device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_POL_SET_ATTR	Can be customized to provide validation for attribute changes. See the discussion on changing the attributes of device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_POL_SET_BRAND	Can be customized to provide validation or other functionality during a brand change. See the discussion on associating devices and brand objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_POL_SET_STATE	Can be customized to provide validation or other functionality for state changes. See the discussion on the discussion on changing the state of a device object in *BRM Developer's Guide*.	Recommended

PCM_OP_DEVICE_POL_ASSOCIATE

Allows 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.

This opcode 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.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_DEVICE_ASSOCIATE standard opcode.

See the discussion on associating service and device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_POL_CREATE

Allows customized validation during device creation. 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.

This opcode 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.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_DEVICE_CREATE standard opcode.

See the discussion on creating device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_POL_DELETE

Allows customized validation during device deletion. 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.

This opcode 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.

By default, this opcode checks whether a device is associated with any services, and if it is, aborts the transaction.

This opcode calls a different opcode to customize device deletion. For example, if PIN_FLD_OBJ_TYPE is **/device/num**, this policy opcode calls the PCM_OP_NUM_POL_DEVICE_DELETE policy opcode to perform the validation checks it contains.

This opcode is called by the PCM_OP_DEVICE_DELETE standard opcode.

See the discussion on deleting device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_POL_SET_ATTR

Allows customized validation of device attribute changes. 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.

This opcode 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.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_DEVICE_SET_ATTR standard opcode.

See the discussion on changing the attributes of device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_POL_SET_BRAND

Allows customized validation of device brand changes. For example, you could limit brand changes to certain device types or situations.

This opcode 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_BRAND to perform the validation checks it contains.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_DEVICE_SET_BRAND standard opcode.

See the discussion on associating devices and brand objects in *BRM Developer's Guide*.

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.

This opcode 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.

This opcode is a hook provided to facilitate customization.

This opcode is called by the PCM_OP_DEVICE_SET_STATE standard opcode.

See the discussion on changing the state of a device object in *BRM Developer's Guide*.

Device FM Standard Opcodes

The opcodes listed in Table 1–27 run device management processes.

Header File

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

Opcode Index

Table 1–27 Device FM Standard Opcodes

Opcode	Description	Use
PCM_OP_DEVICE_ASSOCIATE	Associates services with **/device** objects, or disassociates services from **/device** objects. See the discussion on associating service and device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_CREATE	Creates a new **/device** object. See the discussion on creating device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_DELETE	Deletes a **/device** object. See the discussion on deleting device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_SET_ATTR	Sets attribute values for a **/device** object. See the discussion on changing the attributes of device objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_SET_BRAND	Sets the brand for a **/device** object. See the discussion on associating devices and brand objects in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_SET_STATE	Sets the state for a **/device** object. See the discussion on changing the state of a device object in *BRM Developer's Guide*.	Recommended
PCM_OP_DEVICE_UPDATE	Changes any combination of attribute values, brand, and state of a device.	Recommended

PCM_OP_DEVICE_ASSOCIATE

Associates or disassociates **/service** objects with /device objects.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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_ASSOCIATE, which in turn calls PCM_OP_IP_POL_DEVICE_ASSOCIATE.

Note: You specify which services can be associated with a particular device type and brand by using the **pin_device_permit_map** file. See the discussion on defining device-to-service associations in *BRM Developer's Guide*.

See the discussion on associating service and device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_CREATE

Creates a **/device** object of the type specified in the input flist.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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.

See the discussion on creating device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_DELETE

Deletes a **/device** object.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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.

See the discussion on deleting device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_SET_ATTR

Changes the attributes for a **/device** object.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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.

Note: You cannot use PCM_OP_DEVICE_SET_ATTR alone to change the brand association, device state, or service association. If the input flist includes these fields, they are ignored.

See the discussion on changing the attributes of device objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_SET_BRAND

Changes the brand association of the device.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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_BRAND, which in turn calls PCM_OP_APN_POL_DEVICE_SET_BRAND.

See the discussion on associating devices and brand objects in *BRM Developer's Guide*.

PCM_OP_DEVICE_SET_STATE

Sets the state for a **/device** object.

The validity of each device state change is checked against the **/config/device_state** object for the device type and brand.

This opcode is called by Customer Center and other BRM device managers, and may be called by a custom device-creation application that you create.

This opcode 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.

Important: This opcode uses the event notification feature. Before using this opcode, you must configure event notification for device management. See the discussion on configuring event notification for Device Management in *BRM Developer's Guide*.

See the discussion on changing the state of a device object in *BRM Developer's Guide*.

PCM_OP_DEVICE_UPDATE

Changes any combination of attribute values, brand, and state of a device in a single transaction.

This opcode is called by BRM GUI applications and BRM FMs that modify device characteristics.

This opcode calls these opcodes to perform validation checks before committing any changes:

- PCM_OP_DEVICE_POL_SET_ATTR
- PCM_OP_DEVICE_POL_SET_BRAND
- PCM_OP_DEVICE_POL_SET_STATE

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

Example 1–152 Sample Input Flist

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip -1 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "Testnap"
0 PIN_FLD_ARGS         ARRAY [0]
1   PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 171904 1
1   PIN_FLD_DESCR        STR [0] "TST"
1   PIN_FLD_STATE_ID    INT [0] 4
0 PIN_FLD_ARGS         ARRAY [1]
1   PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 171880 1
1   PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 171856 1
1   PIN_FLD_DESCR        STR [0] "TST1"
1   PIN_FLD_STATE_ID    INT [0] 4
0 PIN_FLD_ARGS         ARRAY [2]
1   PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 171856 1
1   PIN_FLD_DESCR        STR [0] "TST2 QA3"
1   PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 171856 1
```

Example 1–153 Sample Output Flist

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /device/ip -1 0
0 PIN_FLD_RESULTS ARRAY [0]
1   PIN_FLD_POID POID [0] 0.0.0.1 /device/ip 171904 0
0 PIN_FLD_RESULTS ARRAY [1]
1   PIN_FLD_POID POID [0] 0.0.0.1 /device/ip 171880 0
0 PIN_FLD_RESULTS ARRAY [2]
1   PIN_FLD_POID POID [0] 0.0.0.1 /device/ip 171856 0
```

Email Data Manager Opcodes

The Email Data Manager opcodes listed in Table 1–28 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.

Header File

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

Opcode Index

Table 1–28 Email Data Manager Base Opcodes

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

PCM_OP_CREATE_OBJ

Provides a platform-independent interface to **dm_email** for sending one or more email attachments.

PCM_OP_DELIVERY_MAIL_SENDMSGS calls this opcode.

PCM_OP_DELIVERY_MAIL_SENDMSGS

Queries the **pin.conf** file for the location of the Email DM, and ensures the data in the PIN_FLD_MESSAGES array is valid.

This opcode is called by PCM_OP_ACT_POL_EVENT_NOTIFY and PCM_OP_CUST_POL_POST_COMMIT.

Email Manager FM Opcodes

The opcodes listed in Table 1–29 are used to authenticate the email login and authorize the delivery of an incoming email message.

Opcode Index

Table 1–29 Email Manager FM Opcodes

Opcode	Description	Use
PCM_OP_MAIL_DELIV_VERIFY	Authorizes the delivery of an incoming email message. See the discussion on customizing email login authorization in *BRM Email Manager*.	Recommended
PCM_OP_MAIL_LOGIN_VERIFY	Authorizes a **/service/email** user to send and receive messages. See the discussion on customizing email delivery authorization in *BRM Email Manager*.	Recommended

PCM_OP_MAIL_DELIV_VERIFY

Authorizes the delivery of an incoming email message to a user's mail queue. The default check is for an active service status.

See the discussion on customizing email delivery authorization in *BRM Email Manager*.

PCM_OP_MAIL_LOGIN_VERIFY

Authorizes an email user to send and read messages. The default checks for service status, password, and available credit balance greater than or equal to 0.

See the discussion on customizing email login authorization in *BRM Email Manager*.

Filter Set FM Standard Opcodes

This document describes the filter set opcodes listed in Table 1–30. These opcodes support BRM Pricing Center in providing separate products and discounts to the different market segments of your customer base. These opcodes allow you to divide your customers into market segments by filtering them based on criteria that you set in Pricing Center.

For information about using filters sets, see the discussion on using filter sets to apply system products and discounts in *BRM Configuring Pipeline Rating and Discounting*.

Header File

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

Opcode Index

Table 1–30 Filter Set FM Standard Opcodes

Opcode	Description	Use
PCM_OP_FILTER_SET_CREATE	Creates a new **/filter_set/product** object. See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_FILTER_SET_DELETE	Deletes a **/filter_set/product** object. See the discussion on deleting filter sets in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_FILTER_SET_UPDATE	Modifies a **/filter_set/product** object. See the discussion on updating filter sets in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_FILTER_SET_CREATE

Creates **/filter_set/product** objects, which store the list of system products and discounts that belong to a particular filter set. This opcode is called directly by Pricing Center.

See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.

PCM_OP_FILTER_SET_DELETE

Deletes **/filter_set/product** objects. This opcode is called directly by Pricing Center.

See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.

PCM_OP_FILTER_SET_UPDATE

Modifies the following data in **/filter_set/product** objects:

- The filter criteria
- The list of applicable system products and discounts
- The validity period

This opcode is called directly by Pricing Center.

See the discussion on updating filter sets in *BRM Setting Up Pricing and Rating*.

General Ledger FM Policy Opcodes

Use the opcodes in Table 1–31 to customize the data in exported G/L reports.

Header File

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

Opcode Index

Table 1–31 General Ledger FM Policy Opcodes

Opcode	Description	Use
PCM_OP_GL_POL_EXPORT_GL	Customizes the data in the exported G/L reports. See the discussion on customizing G/L reports for export in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_GL_POL_PRE_UPDATE_JOURNAL	Customizes data before it is written into **/journal** objects.	Recommended

PCM_OP_GL_POL_EXPORT_GL

Allows customization of data in the exported G/L reports.

The **pin_ledger_report** utility calls this policy opcode in **-export** mode after it generates a G/L report but before it exports the G/L report data to an XML file.

See the discussion on customizing G/L reports for export in *BRM Configuring and Running Billing*.

PCM_OP_GL_POL_PRE_UPDATE_JOURNAL

Allows customization of general ledger data before it is recorded into **/journal** objects.

General Ledger FM Standard Opcodes

The opcode listed in Table 1–32 are used to calculate account information and create **/ledger_report** objects.

Header File

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

Opcode Index

Table 1–32 General Ledger FM Standard Opcode

Opcode	Description	Use
PCM_OP_GL_LEDGER_REPORT	Calculates account information and creates **/ledger_report** object See the discussion on how BRM stores general ledger reports in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_GL_LEDGER_REPORT

Creates **/ledger_report** objects, which store general ledger reports. This opcode is called directly by the **pin_ledger_report** utility.

When the PCM_OPFLG_READ_RESULT flag is set, the opcode returns the entire contents of the **/ledger_report** object.

See the discussion on how BRM stores general ledger reports in *BRM Configuring and Running Billing*.

GPRS Manager 3.0 FM Policy Opcodes

Use the opcode in Table 1–33 to customize the GPRS service extensions.

Header File

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

Opcode Index

Table 1–33 GPRS Manager 3.0 FM Policy Opcode

Opcode	Description	Use
PCM_OP_GPRS_POL_APPLY_PARAMETER	Customizes the GPRS service extensions. See the discussion on updating custom GPRS service fields in *BRM Telco Integration*.	Recommended

PCM_OP_GPRS_POL_APPLY_PARAMETER

Allows customization of GPRS service extensions.

This opcode is called by the PCM_OP_GPRS_APPLY_PARAMETER standard opcode.

See the discussion on updating custom GPRS service fields in *BRM Telco Integration*.

GPRS Manager 3.0 FM Standard Opcodes

The opcode in Table 1–34 is used to add GPRS-specific service extensions.

Header File

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

Opcode Index

Table 1–34 GPRS Manager 3.0 FM Standard Opcode

Opcode	Description	Use
PCM_OP_GPRS_APPLY_PARAMETER	Adds GPRS-specific service extensions. See the discussion on associating APN and QoS pairs with GPRS services in *BRM Telco Integration*.	Recommended

PCM_OP_GPRS_APPLY_PARAMETER

Reads the service extensions from the input flist and adds corresponding GPRS service values.

This opcodes calls the PCM_OP_GPRS_POL_APPLY_PARAMETER policy opcode to apply any customizations.

GPRS AAA Manager FM Helper Policy Opcodes

The opcodes listed in Table 1–35 are called by the Services Framework AAA standard opcodes to perform service-specific operations, such as building search templates and aggregating GPRS data.

For more information about GPRS AAA Manager, see the discussion on performing AAA for prepaid GPRS services in *BRM Telco Integration*.

About Helper Opcodes

Helper opcodes are called during one of these stages in the execution of a Services Framework AAA FM standard opcode:

- SEARCH_SESSION
- PREP_INPUT
- VALIDATE_LIFECYCLE
- POST_PROCESS
- ACC_ON_OFF_SEARCH

You can configure Services Framework AAA opcodes to call the helper opcodes by using the **load_aaa_config_opcodemap_tcf** utility. See the discussion on configuring Services Framework to call helper opcodes in *BRM Telco Integration*.

Header File

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

Opcode Index

Table 1–35 GPRS AAA Manager FM Helper Policy Opcodes

Opcode	Description	Use
PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH	Builds search templates for finding **/active_session/telco/gprs** objects. See the discussion on building search templates for GPRS active session objects in *BRM Telco Integration*.	Recommended
PCM_OP_GPRS_AAA_POL_AUTHORIZE_PREP_INPUT	Prepares input flists that can be used for authorizing a GPRS session. See the discussion on preparing GPRS-specific flists for authorization in *BRM Telco Integration*.	Recommended
PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT	Prepares flists that can be used for reauthorizing a GPRS session. See the discussion on preparing GPRS-specific flists for reauthorization in *BRM Telco Integration*.	Recommended

Table 1–35 (Cont.) GPRS AAA Manager FM Helper Policy Opcodes

Opcode	Description	Use
PCM_OP_GPRS_AAA_POL_SEARCH_SESSION	Builds search templates for finding **/active_session/telco/gprs** or **/event/session/telco/gprs** objects. See the discussion on building search templates for GPRS session objects in *BRM Telco Integration*.	Recommended
PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT	Prepares input flists that can be used for ending a prepaid GPRS session. See the discussion on preparing GPRS-specific flists for ending sessions in *BRM Telco Integration*.	Recommended
PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT	Prepares input flists that can be used for updating a prepaid GPRS session. See the discussion on preparing GPRS-specific flists for updating sessions in *BRM Telco Integration*.	Recommended

PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH

Builds a search template that can be used to find **/active_session/telco/gprs** objects.

This opcode is called by the PCM_OP_TCF_AAA_ACCOUNTING_OFF standard opcode when processing **/service/telco/gprs** events.

See the discussion on building search templates for GPRS active session objects in *BRM Telco Integration*.

PCM_OP_GPRS_AAA_POL_AUTHORIZE_PREP_INPUT

Aggregates GPRS data and prepares an input flist for authorizing prepaid GPRS sessions.

This opcode is called by the PCM_OP_TCF_AAA_AUTHORIZE standard opcode when processing additional service types or to change which helper opcodes are called.

See the discussion on preparing GPRS-specific flists for authorization in *BRM Telco Integration*.

PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT

Aggregates GPRS data and prepares an input flist for reauthorizing prepaid GPRS sessions.

This opcode is called by the PCM_OP_TCF_AAA_REAUTHORIZE standard opcode when processing **/service/telco/gprs** events.

See the discussion on preparing GPRS-specific flists for reauthorization in *BRM Telco Integration*.

PCM_OP_GPRS_AAA_POL_SEARCH_SESSION

Builds search templates for finding **/active_session/telco/gprs** or **/event/session/telco/gprs** objects.

This opcode is called by the PCM_OP_TCF_AAA_REAUTHORIZE standard opcode when processing **/service/telco/gprs** events.

See the discussion on building search templates for GPRS session objects in *BRM Telco Integration*.

Example 1–154 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gprs -1 0
0 PIN_FLD_MSID                  STR [0] "380-20060918-201727-0-16576-1-blrhpdv3"
0 PIN_FLD_REQ_MODE             ENUM [0] 1
0 PIN_FLD_PROGRAM_NAME          STR [0] "testnap"
0 PIN_FLD_ORIGIN_NETWORK        STR [0] "Portal"
0 PIN_FLD_DIRECTION            ENUM [0] 0
0 PIN_FLD_OBJ_TYPE              STR [0] "gprs"
0 PIN_FLD_AUTHORIZATION_ID      STR [0] "GPRS002"
0 PIN_FLD_EXTENDED_INFO   SUBSTRUCT [0] allocated 20, used 1
1   PIN_FLD_GPRS_INFO     SUBSTRUCT [0] allocated 20, used 4
2     PIN_FLD_GGSN_ADDRESS      STR [0] "gprs1"
2     PIN_FLD_SGSN_ADDRESS      STR [0] "gprs1"
2     PIN_FLD_CELL_ID           STR [0] "gprs1"
2     PIN_FLD_APN               STR [0] "gprs1"
0 PIN_FLD_SERVICE_OBJ          POID [0] 0.0.0.1 /service/telco/gprs 561208 6
0 PIN_FLD_ACCOUNT_OBJ          POID [0] 0.0.0.1 /account 564952 0
0 PIN_FLD_OPCODE                INT [0] 4003
```

Example 1–155 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_FLAGS                 INT [0] 256
0 PIN_FLD_AUTHORIZATION_ID      STR [0] "GPRS002"
0 PIN_FLD_ARGS                ARRAY [1] allocated 20, used 1
1   PIN_FLD_ACTIVE_SESSION_ID   STR [0] "GPRS002"
0 PIN_FLD_INDEX_NAME            STR [0] "active_session_active_id_i"
0 PIN_FLD_ARGS                ARRAY [2] allocated 20, used 1
1   PIN_FLD_GPRS_INFO     SUBSTRUCT [0] allocated 20, used 1
2     PIN_FLD_APN               STR [0] "gprs1"
0 PIN_FLD_ARGS                ARRAY [3] allocated 20, used 1
1   PIN_FLD_GPRS_INFO     SUBSTRUCT [0] allocated 20, used 1
2     PIN_FLD_GGSN_ADDRESS      STR [0] "gprs1"
0 PIN_FLD_ARGS                ARRAY [4] allocated 20, used 1
1   PIN_FLD_GPRS_INFO     SUBSTRUCT [0] allocated 20, used 1
2     PIN_FLD_SGSN_ADDRESS      STR [0] "gprs1"
0 PIN_FLD_TEMPLATE              STR [0] "select X from /active_session/telco/gprs
                                         where F1 = V1  and F2 = V2 and F3 = V3
and F4 = V4 "
0 PIN_FLD_RESULTS             ARRAY [0] allocated 20, used 0
```

PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT

Aggregates GPRS data and prepares an input flist for ending a prepaid GPRS session.

This opcode is called by the PCM_OP_TCF_AAA_ACCOUNTING standard opcode when processing **/service/telco/gprs** events.

See the discussion on preparing GPRS-specific flists for ending sessions in *BRM Telco Integration*.

Example 1–156 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /service/telco/gprs -1 0
0 PIN_FLD_MSID                     STR [0]
"930-20060913-135601-0-27241-1-blrhpdv3"
0 PIN_FLD_REQ_MODE                ENUM [0] 2
0 PIN_FLD_SESSION_STOP_INDICATOR  ENUM [0] 1
0 PIN_FLD_PROGRAM_NAME             STR [0] "testnap"
0 PIN_FLD_AUTHORIZATION_ID         STR [0] "gprs06"
0 PIN_FLD_DIRECTION               ENUM [0] 0
0 PIN_FLD_BYTES_UPLINK         DECIMAL [0] 5120
0 PIN_FLD_BYTES_DOWNLINK       DECIMAL [0] 5120
0 PIN_FLD_OBJ_TYPE                 STR [0] "gprs"
0 PIN_FLD_EXTENDED_INFO      SUBSTRUCT [0] allocated 20, used 1
1   PIN_FLD_GPRS_INFO        SUBSTRUCT [0] allocated 20, used 4
2     PIN_FLD_GGSN_ADDRESS         STR [0] "gprs11"
2     PIN_FLD_SGSN_ADDRESS         STR [0] "gprs11"
2     PIN_FLD_CELL_ID              STR [0] "gprs11"
2     PIN_FLD_APN                  STR [0] "gprs11"
```

Example 1–157 Sample Output Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /event/session/telco/gprs 546679
0
0 PIN_FLD_AUTHORIZATION_ID       STR [0] "gprs06"
0 PIN_FLD_ACCOUNT_OBJ           POID [0] 0.0.0.1 /account 521419 0
0 PIN_FLD_SERVICE_OBJ           POID [0] 0.0.0.1 /service/telco/gprs 523723 8
0 PIN_FLD_RATING_STATUS         ENUM [0] 1
```

PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT

Aggregates GPRS data and prepares an input flist for updating a prepaid GPRS session.

This opcode is called by the PCM_OP_TCF_AAA_REAUTHORIZE standard opcode when processing **/service/telco/gprs** events.

See the discussion on preparing GPRS-specific flists for updating sessions in *BRM Telco Integration*.

Example 1–158 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/gprs -1 0
0 PIN_FLD_QUANTITY          DECIMAL [0] 100
0 PIN_FLD_MSID                  STR [0] "930-20060913-135601-0-27241-1-blrhpdv3"
0 PIN_FLD_REQ_MODE             ENUM [0] 2
0 PIN_FLD_PROGRAM_NAME          STR [0] "testnap"
0 PIN_FLD_ORIGIN_NETWORK        STR [0] "Portal"
0 PIN_FLD_BYTES_UPLINK      DECIMAL [0] 2048
0 PIN_FLD_DIRECTION            ENUM [0] 0
0 PIN_FLD_BYTES_DOWNLINK    DECIMAL [0] 3072
0 PIN_FLD_OBJ_TYPE              STR [0] "gprs"
0 PIN_FLD_AUTHORIZATION_ID      STR [0] "GPRS002"
0 PIN_FLD_EXTENDED_INFO    SUBSTRUCT [0] allocated 20, used 1
1   PIN_FLD_GPRS_INFO      SUBSTRUCT [0] allocated 20, used 4
2     PIN_FLD_GGSN_ADDRESS      STR [0] "gprs1"
2     PIN_FLD_SGSN_ADDRESS      STR [0] "gprs1"
2     PIN_FLD_CELL_ID           STR [0] "gprs1"
2     PIN_FLD_APN               STR [0] "gprs1"
```

Example 1–159 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /active_session/telco/gprs 571314
0
0 PIN_FLD_AUTHORIZATION_ID      STR [0] "GPRS002"
0 PIN_FLD_ACCOUNT_OBJ          POID [0] 0.0.0.1 /account 521419 0
0 PIN_FLD_SERVICE_OBJ          POID [0] 0.0.0.1 /service/telco/gprs 523723 8
```

GPRS AAA Manager FM Policy Opcodes

Use the opcode in Table 1–36 to customize generation of a unique authorization ID.

Header File

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

Error Handling

The GPRS AAA Manager FM policy opcodes check if ebuf is set before performing each step. If the ebuf is set, processing stops and an ebuf exception is passed to the caller.

Opcode Index

Table 1–36 GPRS AAA Manager FM Policy Opcode

Opcode	Description	Use
PCM_OP_GPRS_AAA_POL_AUTHORIZE	Generates a unique authorization ID if one does not already exist. See the discussion on customizing GPRS authorization IDs in *BRM Telco Integration*.	Recommended

PCM_OP_GPRS_AAA_POL_AUTHORIZE

Generates a unique authorization ID for a GPRS session if one is not passed in the input flist. This opcode is called by the PCM_OP_GPRS_AAA_POL_SEARCH_SESSION helper opcode when processing GPRS authorization requests.

By default, this opcode generates IDs that use the following format:

APN - GGSN_Address - SGSN_Address - START_T

However, you can customize this opcode to use another ID format.

This opcode is called by the PCM_OP_TCF_AAA_AUTHORIZE standard opcode when processing additional service types or to change which helper opcodes are called.

See the discussion on customizing GPRS authorization IDs in *BRM Telco Integration*.

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /service/telco/gprs -1 0
0 PIN_FLD_MSID                 STR [0] "693-20060808-123322-0-2050-1-blr-insat"
0 PIN_FLD_PROGRAM_NAME         STR [0] "testnap"
0 PIN_FLD_QUANTITY         DECIMAL [0] 100
0 PIN_FLD_START_T           TSTAMP [0] (1155020599)
0 PIN_FLD_ORIGIN_NETWORK       STR [0] "Portal"
0 PIN_FLD_OPCODE               INT [0] 4002
0 PIN_FLD_OBJ_TYPE             STR [0] "gprs"
0 PIN_FLD_REQ_MODE            ENUM [0] 4
0 PIN_FLD_EXTENDED_INFO  SUBSTRUCT [0]
1   PIN_FLD_GPRS_INFO    SUBSTRUCT [0]
2     PIN_FLD_APN              STR [0] "test1"
2     PIN_FLD_GGSN_ADDRESS     STR [0] "test1"
2     PIN_FLD_SGSN_ADDRESS     STR [0] "test1"
```

Example 1–160 Sample Output Flist

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /active_session/telco/gprs 429163 0
0 PIN_FLD_EXPIRATION_T      TSTAMP [0] (1155189354) Thu Aug 10 11:25:54 2006
0 PIN_FLD_QUANTITY         DECIMAL [0] 50
0 PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.1 /service/telco/gprs 427563 6
0 PIN_FLD_RESERVATION_OBJ     POID [0] 0.0.0.1 /reservation/active 426603 0
0 PIN_FLD_BAL_GRP_OBJ         POID [0] 0.0.0.1 /balance_group 429099 1
0 PIN_FLD_BALANCES           ARRAY [840] allocated 20, used 1
1   PIN_FLD_AMOUNT         DECIMAL [0] 0.05
0 PIN_FLD_RESULT              ENUM [0] 1
0 PIN_FLD_RATING_STATUS       ENUM [0] 0
0 PIN_FLD_AUTHORIZATION_ID     STR [0] "test1-1155020599-test1-test1"
0 PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 429003 0
```

Group FM Standard Opcodes

The opcodes listed in Table 1–37 create and delete account groups and account group members.

Header File

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

Opcode Index

Table 1–37 Group FM Standard Opcodes

Opcode	Description	Use
PCM_OP_GROUP_ADD_MEMBER	Adds members to a group. See the discussion on adding members to a group in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_GROUP_CREATE_GROUP	Creates a new group object. See the discussion on creating a group in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_GROUP_DELETE_GROUP	Deletes an existing group object. See the discussion on deleting a group in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_GROUP_DELETE_MEMBER	Deletes members from a group. See the discussion on deleting members from a group in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_GROUP_SET_PARENT	Sets the parent object of a group. See the discussion on setting a group parent in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_GROUP_UPDATE_INHERITED	Updates the inheritance fields of an existing group. See the discussion on updating the inheritance fields in a group in *BRM Managing Accounts Receivable*.	Limited

PCM_OP_GROUP_ADD_MEMBER

Adds one or more members to an existing group.

See the discussion on adding members to a group in *BRM Managing Accounts Receivable*.

PCM_OP_GROUP_CREATE_GROUP

Creates a new group object.

See the discussion on creating a group in *BRM Managing Accounts Receivable*.

PCM_OP_GROUP_DELETE_GROUP

Deletes an existing **/group** object from the database.

See the discussion on deleting a group in *BRM Managing Accounts Receivable*.

PCM_OP_GROUP_DELETE_MEMBER

Deletes one or more members from an existing group.

See the discussion on deleting members from a group in *BRM Managing Accounts Receivable*.

PCM_OP_GROUP_SET_PARENT

Sets parent object of a group.

See the discussion on setting a group parent in *BRM Managing Accounts Receivable*.

PCM_OP_GROUP_UPDATE_INHERITED

Updates the inheritance fields of an existing group.

See the discussion on updating the inheritance fields in a group in *BRM Managing Accounts Receivable*.

GSM AAA Manager FM Helper Policy Opcodes

The opcodes listed in Table 1–38 are called by the Services Framework AAA standard opcodes to perform service-specific operations, such as building search templates and aggregating GSM data.

For more information about GSM AAA Manager, see the discussion on performing AAA for prepaid GSM services in *BRM Telco Integration*.

About Helper Opcodes

Helper opcodes are called during one of these stages in the execution of a Services Framework AAA FM standard opcode:

- SEARCH_SESSION
- PREP_INPUT
- VALIDATE_LIFECYCLE
- ACC_ON_OFF_SEARCH
- POST_PROCESS

You can configure Services Framework AAA opcodes to call the helper opcodes by using the **load_aaa_config_opcodemap_tcf** utility. See the discussion on configuring Services Framework to call helper opcodes in *BRM Telco Integration*.

Header File

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

Opcode Index

Table 1–38 GSM AAA Manager FM Helper Policy Opcodes

Opcode	Description	Use
PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH	Builds search templates for finding **/active_session/telco/gsm** objects. See the discussion on building search templates for GSM active session objects in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_POL_AUTHORIZE_PREP_INPUT	Builds flists for authorizing GSM sessions. See the discussion on preparing GSM-specific input flists for authorization in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_POL_POST_PROCESS	Aggregates GSM data returned from the update and reauthorization processes. See the discussion on aggregating return GSM data in *BRM Telco Integration*.	Recommended

Table 1–38 (Cont.) GSM AAA Manager FM Helper Policy Opcodes

Opcode	Description	Use
PCM_OP_GSM_AAA_POL_REAUTHORIZE_PREP_INPUT	Builds flists for reauthorizing GSM sessions. See the discussion on preparing GSM-specific input flists for reauthorization in *BRM Telco Integration*.	Recommended
PCM_OP_GPRS_AAA_POL_SEARCH_SESSION	Builds search templates for finding **/active_session/telco/gsm** or **/event/session/telco/gsm** objects. See the discussion on building search templates for GSM session objects in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_POL_STOP_ACCOUNTING_PREP_INPUT	Builds flists for ending prepaid GSM accounting sessions. See the discussion on preparing GSM-specific input flists for stopping accounting sessions in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT	Builds flists for updating existing prepaid GSM accounting sessions. See the discussion on the discussion on preparing GSM-specific input flists for updating accounting sessions in *BRM Telco Integration*.	Recommended

PCM_OP_GSM_AAA_POL_ACC_ON_OFF_SEARCH

Builds a search template that can be used to find **/active_session/telco/gsm** objects.

This opcode is called by the PCM_OP_TCF_AAA_ACCOUNTING_OFF standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on building search templates for GSM active session objects in *BRM Telco Integration*.

Example 1–161 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /service/telco/gsm/telephony
0 PIN_FLD_ORIGIN_NETWORK           STR [0] "Network 1234"
0 PIN_FLD_START_T                TSTAMP [0] (1111737600) Fri Mar 25 00:00:00 2005
```

Example 1–162 Sample Output Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /service/telco/gsm/telephony
0 PIN_FLD_RESULTS              ARRAY [0] allocated 2, used 2
0 PIN_FLD_ARGS                 ARRAY [0] allocated 3, used 3
1   PIN_FLD_ORIGIN_NETWORK       STR [0] "Network 1234"
1   PIN_FLD_FLAGS               FLAG [0] ORIGIN_NETWORK
1   PIN_FLD_TEMPLATE             STR [0] "select X from /active_
session/telco/gsm/telephony
                                         where ORIGIN_NETWORK = Network 1234"
```

PCM_OP_GSM_AAA_POL_AUTHORIZE_PREP_INPUT

Aggregates GSM data and prepares an input flist for authorizing prepaid GSM sessions.

This opcode is called by the PCM_OP_TCF_AAA_ACCOUNTING_OFF standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on preparing GSM-specific input flists for authorization in *BRM Telco Integration*.

PCM_OP_GSM_AAA_POL_POST_PROCESS

Aggregates data returned from the update and reauthorization processes.

This opcode is called by the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on aggregating return GSM data in *BRM Telco Integration*.

PCM_OP_GSM_AAA_POL_REAUTHORIZE_PREP_INPUT

Aggregates GSM data and prepares an flist for the reauthorization process.

This opcode is called by the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on preparing GSM-specific input flists for reauthorization in *BRM Telco Integration*.

PCM_OP_GSM_AAA_POL_SEARCH_SESSION

Builds search templates for finding **/active_session/telco/gsm** or **/event/session/telco/gsm** objects.

This opcode is called by the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on building search templates for GSM session objects in *BRM Telco Integration*.

Example 1–163 Sample Input Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /active_session/telco/gsm
0 PIN_FLD_PROGRAM_NAME            STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID        STR [0] "24874654"
0 PIN_FLD_DIRECTION              ENUM [0] 0
```

Example 1–164 Sample Output Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_ARGS                  ARRAY [1] allocated 100, used 1
1   PIN_FLD_ACTIVE_SESSION_ID     STR [0] "00101771211133403461100004"
0 PIN_FLD_INDEX_NAME              STR [0] "active_session_active_id_i"
0 PIN_FLD_FLAGS                   INT [0] 256
0 PIN_FLD_TEMPLATE                STR [0] "select X from /active_session/telco/gsm
where
                                           F1 = V1 "
0 PIN_FLD_RESULTS               ARRAY [0] NULL array ptr
```

PCM_OP_GSM_AAA_POL_STOP_ACCOUNTING_PREP_INPUT

Aggregates GSM data and prepares flists for ending a prepaid GSM session.

This opcode is called by the PCM_OP_TCF_AAA_STOP_ACCOUNTING standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on preparing GSM-specific input flists for stopping accounting sessions in *BRM Telco Integration*.

PCM_OP_GSM_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT

Aggregates GSM data and prepares flists for updating an existing prepaid GSM session.

This opcode is called by the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE standard opcode when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

See the discussion on preparing GSM-specific input flists for updating accounting sessions in *BRM Telco Integration*.

GSM AAA Manager FM Policy Opcodes

Use the opcode in Table 1–39 to customize generation of a unique authorization ID for a GSM session.

Header File

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

Error Handling

The GSM AAA Manager FM policy opcodes check if ebuf is set before performing each step. If the ebuf is set, processing stops and an ebuf exception is passed to the caller.

Opcode Index

Table 1–39 GSM AAA Manager FM Policy Opcode

Opcode	Description	Use
PCM_OP_GSM_AAA_POL_AUTHORIZE	Generates a unique authorization ID if one does not already exist. See the discussion on customizing GSM authorization IDs in *BRM Telco Integration*.	Recommended

PCM_OP_GSM_AAA_POL_AUTHORIZE

Generates a unique authorization ID for a GSM session if one is not passed in the input flist.

This opcode is called by the PCM_OP_GSM_AAA_AUTHORIZE and PCM_OP_GSM_AUTHORIZE standard opcodes.

See the discussion on customizing GSM authorization IDs in *BRM Telco Integration*.

Example 1–165 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /service/telco/gsm/telephony
0 PIN_FLD_MSID              STR [0] "9283472938"
0 PIN_FLD_PROGRAM_NAME      STR [0] "sample_act"
0 PIN_FLD_DIRECTION        ENUM [0] 0
0 PIN_FLD_DIALED_NUMBER     STR [0] "4085551212"
```

Example 1–166 Sample Output Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /service/telco/gsm/telephony
0 PIN_FLD_MSID              STR [0] "9283472938"
0 PIN_FLD_PROGRAM_NAME      STR [0] "sample_act"
0 PIN_FLD_DIRECTION        ENUM [0] 0
0 PIN_FLD_DIALED_NUMBER     STR [0] "4085551212"
0 PIN_FLD_AUTHORIZATION_ID  STR [0] "2398472934398759 403980"
```

GSM AAA Manager FM Standard Opcodes

The opcodes listed in Table 1–40 process AAA requests from external networks.

Header File

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

Error Handling

The GSM AAA Manager FM standard opcodes check if **ebuf** is set before performing each step. If the **ebuf** is set, processing stops and an **ebuf** exception is passed to the caller.

Opcode Index

Table 1–40 GSM AAA Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_GSM_AAA_ACCOUNTING_OFF	Closes all open sessions when the network shuts down or encounters a problem. See the discussion on closing prepaid GSM sessions when the external network shuts down in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_ACCOUNTING_ON	Closes all open sessions when the external network restarts. See the discussion on closing prepaid GSM sessions when the external network restarts in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_AUTHENTICATE	Authenticates users for GSM services. See the discussion on authenticating users for GSM services in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_AUTHORIZE	Authorizes GSM sessions. See the discussion on authorizing GSM services in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_CANCEL_AUTHORIZATION	Cancels an authorization and releases any reserved resources. See the discussion on canceling authorization for GSM services in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_REAUTHORIZE	Reauthorizes GSM accounting sessions. See the discussion on reauthorizing GSM sessions in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_START_ACCOUNTING	Starts GSM accounting sessions. See the discussion on starting prepaid GSM sessions in *BRM Telco Integration*.	Recommended

Table 1–40 (Cont.) GSM AAA Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_GSM_AAA_STOP_ACCOUNTING	Ends GSM accounting sessions. See the discussion on ending prepaid GSM sessions in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_UPDATE_ACCOUNTING	Updates information about an existing GSM accounting session. See the discussion on updating a prepaid GSM session in *BRM Telco Integration*.	Recommended
PCM_OP_GSM_AAA_UPDATE_AND_REAUTHORIZE	Updates information about an existing accounting session and then reauthorizes usage. See the discussion on updating and reauthorizing GSM sessions in *BRM Telco Integration*.	Recommended

PCM_OP_GSM_AAA_ACCOUNTING_OFF

Closes all open GSM sessions when the external network shuts down or encounters a severe problem.

See the discussion on closing prepaid GSM sessions when the external network shuts down in *BRM Telco Integration*.

Example 1–167 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /service/telco/gsm 3215649 11
0 PIN_FLD_ORIGIN_NETWORK      STR [0] "Sample Network"
0 PIN_FLD_ACC_FLAG            INT [0] 0
0 PIN_FLD_START_T          TSTAMP [0] (1095379771) Thu Sep 16 17:09:31 2004
0 PIN_FLD_END_T            TSTAMP [0] (1095383091) Thu Sep 16 18:04:51 2004
0 PIN_FLD_TERMINATE_CAUSE    ENUM [0] 10
```

Example 1–168 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /service/telco/gsm 3215649 11
```

PCM_OP_GSM_AAA_ACCOUNTING_ON

Closes all open GSM sessions when the external network restarts.

See the discussion on closing prepaid GSM sessions when the external network restarts in *BRM Telco Integration*.

Example 1–169 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /service/telco/gsm 3215649 11
0 PIN_FLD_ORIGIN_NETWORK      STR [0] "Sample Network"
0 PIN_FLD_ACC_FLAG            INT [0] 0
0 PIN_FLD_START_T          TSTAMP [0] (1095379771) Thu Sep 16 17:09:31 2004
0 PIN_FLD_END_T            TSTAMP [0] (1095383091) Thu Sep 16 18:04:51 2004
0 PIN_FLD_TERMINATE_CAUSE    ENUM [0] 10
```

Example 1–170 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /service/telco/gsm 3215649 11
```

PCM_OP_GSM_AAA_AUTHENTICATE

Authenticates users for GSM services.

See the discussion on authenticating users for GSM services in *BRM Telco Integration*.

Example 1–171 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /service/telco/gsm -1 0
0 PIN_FLD_PROGRAM_NAME   STR [0] "sample_act"
0 PIN_FLD_MSID           STR [0] "15305551234"
0 PIN_FLD_PASSWORD       STR [0] "alkdsjfopi55a7e6ae4r"
```

Example 1–172 Sample Output Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /service/telco/gsm 34564168 51
0 PIN_FLD_MSID           STR [0] "15305551234"
0 PIN_FLD_RESULT        ENUM [0] 0
```

PCM_OP_GSM_AAA_AUTHORIZE

Authorizes users to access GSM services.

See the discussion on authorizing GSM services in *BRM Telco Integration*.

Example 1–173 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1
/service/telco/gsm/telephony -1 0
0 PIN_FLD_PROGRAM_NAME                     STR [0] "sample_act"
0 PIN_FLD_MSID                             STR [0] "19145559876"
0 PIN_FLD_AUTHORIZATION_ID                 STR [0] "asd45f898wae654fdsa"
0 PIN_FLD_DIRECTION                       ENUM [0] 0
0 PIN_FLD_ORIGIN_SID                       STR [0] "origin"
```

Example 1–174 Sample Output Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /active_
session/telco/gsm/telephony
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 321684 10
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1
/service/telco/gsm/telephony 2465 1
0 PIN_FLD_AUTHORIZATION_ID                ENUM [0] "asd45f898wae654fdsa"
0 PIN_FLD_RESULT                          ENUM [0] 1
0 PIN_FLD_REASON                          ENUM [0] 0
0 PIN_FLD_BALANCES                       ARRAY [0] allocated 3, used 3
1   PIN_FLD_AMOUNT                     DECIMAL [0] 20.0
1   PIN_FLD_AVAILABLE_RESOURCE_LIMIT   DECIMAL [0] 100.0
1   PIN_FLD_RUM_NAME                       STR [0] "Sample_RUM"
```

PCM_OP_GSM_AAA_CANCEL_AUTHORIZATION

Cancels an existing GSM authorization and releases reserved resources back to the customer's account balance.

See the discussion on canceling authorization for GSM services in *BRM Telco Integration*.

Example 1–175 Sample Input Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /service/telco/gsm/data -1 0
0 PIN_FLD_MSID                   STR [0] "14085559141"
0 PIN_FLD_PROGRAM_NAME           STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID       STR [0] "alkdsjfopi55a7e6ae4r"
```

Example 1–176 Sample Output Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /service/telco/gsm/data 34564168
51
0 PIN_FLD_AUTHORIZATION_ID       STR [0] "alkdsjfopi55a7e6ae4r"
```

PCM_OP_GSM_AAA_REAUTHORIZE

Reauthorizes GSM accounting sessions.

See the discussion on reauthorizing GSM sessions in *BRM Telco Integration*.

Example 1–177 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /service/telco/gsm/sms
0 PIN_FLD_MSID                           STR [0] "14085556548"
0 PIN_FLD_PROGRAM_NAME                   STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID               STR [0] "alkdsjfopi55a7e6ae4r"
0 PIN_FLD_AMOUNT                     DECIMAL [0] 25.0
0 PIN_FLD_DIRECTION                     ENUM [0] 1
```

Example 1–178 Sample Output Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /active_
session/telco/gsm/sms
0 PIN_FLD_ACCOUNT_OBJ                   POID [0] 0.0.0.1 /account 321684 10
0 PIN_FLD_SERVICE_OBJ                   POID [0] 0.0.0.1 /service/telco/gsm/sms
34564168 51
0 PIN_FLD_AUTHORIZATION_ID              ENUM [0] "alkdsjfopi55a7e6ae4r"
0 PIN_FLD_RESULT                        ENUM [0] 1
0 PIN_FLD_REASON                        ENUM [0] 0
0 PIN_FLD_BALANCES                     ARRAY [0] allocated 3, used 3
1   PIN_FLD_AMOUNT                   DECIMAL [0] 25.0
1   PIN_FLD_AVAILABLE_RESOURCE_LIMI  DECIMAL [0] 100
1   PIN_FLD_RUM_NAME                     STR [0] "sample_rum"
```

PCM_OP_GSM_AAA_START_ACCOUNTING

Starts a GSM accounting session.

See the discussion on starting prepaid GSM sessions in *BRM Telco Integration*.

Example 1–179 Sample Input Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /service/telco/gsm/fax -1 0
0 PIN_FLD_MSID                   STR [0] "14085551234"
0 PIN_FLD_PROGRAM_NAME           STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID       STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_AMOUNT             DECIMAL [0] 40.0
0 PIN_FLD_DIRECTION              ENUM [0] 0
```

Example 1–180 Sample Output Flist

```
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /active_session/telco/gsm/fax
24554 11
0 PIN_FLD_AUTHORIZATION_ID       STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_ACCOUNT_OBJ           POID [0] 0.0.0.1 /account 7512687 11
0 PIN_FLD_SERVICE_OBJ           POID [0] 0.0.0.1 /service/telco/gsm/fax 21367 41
```

PCM_OP_GSM_AAA_STOP_ACCOUNTING

Ends a GSM accounting session. This opcode rates any usage and records the event in the BRM database.

See the discussion on ending prepaid GSM sessions in *BRM Telco Integration*.

Example 1–181 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1
/service/telco/gsm/telephony -1 0
0 PIN_FLD_MSID                             STR [0] "14085551234"
0 PIN_FLD_PROGRAM_NAME                     STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID                 STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_AMOUNT                       DECIMAL [0] 40.0
0 PIN_FLD_DIRECTION                       ENUM [0] 0
```

Example 1–182 Sample Output Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1
/event/session/telco/gsm/telephony 24554 1
0 PIN_FLD_AUTHORIZATION_ID                 STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 7512687 11
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1
/service/telco/gsm/telephony 267 1
0 PIN_FLD_BALANCES                      ARRAY [0] allocated 2, used 2
1   PIN_FLD_AMOUNT                     DECIMAL [0] 40.0
1   PIN_FLD_AVAILABLE_RESOURCE_LIMIT   DECIMAL [0] 120.0
```

PCM_OP_GSM_AAA_UPDATE_ACCOUNTING

Updates information about an existing GSM accounting session.

See the discussion on updating a prepaid GSM session in *BRM Telco Integration*.

Example 1–183 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /service/telco/gsm/sms
-1 0
0 PIN_FLD_MSID                             STR [0] "14085551234"
0 PIN_FLD_PROGRAM_NAME                     STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID                 STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_DIRECTION                       ENUM [0] 1
0 PIN_FLD_DIALED_NUMBER                    STR [0] "14085551212"
```

Example 1–184 Sample Output Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /active_
session/telco/gsm/sms 24554 11
0 PIN_FLD_AUTHORIZATION_ID                 STR [0] "aokjgrt457a9w7t8rae2t"
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 2464787 10
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1 /service/telco/gsm/sms
32468 11
```

PCM_OP_GSM_AAA_UPDATE_AND_REAUTHORIZE

Updates the customer's usage data and reauthorizes GSM sessions in one transaction.

See the discussion on updating and reauthorizing GSM sessions in *BRM Telco Integration*.

Example 1–185 Sample Input Flist

```
0 PIN_FLD_POID                           POID [0] 0.0.0.1 /service/telco/gsm/sms -1
0
0 PIN_FLD_MSID                            STR [0] "18065554578"
0 PIN_FLD_PROGRAM_NAME                    STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID                STR [0] "alkdsjfopi55a7e6ae4r"
0 PIN_FLD_AMOUNT                      DECIMAL [0] 25.0
0 PIN_FLD_DIRECTION                      ENUM [0] 1
```

Example 1–186 Sample Output Flist

```
0 PIN_FLD_POID                           POID [0] 0.0.0.1 /active_
session/telco/gsm/sms
0 PIN_FLD_ACCOUNT_OBJ                    POID [0] 0.0.0.1 /account 321684 10
0 PIN_FLD_SERVICE_OBJ                    POID [0] 0.0.0.1 /service/telco/gsm/sms
34564168 51
0 PIN_FLD_AUTHORIZATION_ID               ENUM [0] "alkdsjfopi55a7e6ae4r"
0 PIN_FLD_RESULT                         ENUM [0] 1
0 PIN_FLD_REASON                         ENUM [0] 0
0 PIN_FLD_BALANCES                      ARRAY [0] allocated 3, used 3
1   PIN_FLD_AMOUNT                    DECIMAL [0] 25.0
1   PIN_FLD_AVAILABLE_RESOURCE_LIMI   DECIMAL [0] 100
1   PIN_FLD_RUM_NAME                      STR [0] "sample_rum"
```

GSM Manager FM Policy Opcodes

Use the opcode in Table 1–41 to add custom service attributes.

Header File

Include the **ops/gsm.h** header file in all applications that call this opcode. See the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–41 GSM Manager FM Policy Opcode

Opcode	Description	Use
PCM_OP_GSM_POL_APPLY_PARAMETER	This policy opcode adds custom service attributes.	Recommended

PCM_OP_GSM_POL_APPLY_PARAMETER

Updates the service flist by adding values to customized fields for a service. This policy opcode takes as input the configuration object flist, the service flist, and the inherited information flist from the calling PCM_OP_GSM_APPLY_PARAMETER opcode.

This opcode is called by the PCM_OP_GSM_APPLY_PARAMETER standard opcode to apply any customizations.

See the discussion on modifying policy files in *BRM Telco Integration*.

Customizing the Opcode

By default, this policy opcode returns an empty inherited info flist.

Customization Example

If you added a substruct to a customized service, you can use this opcode to fill in the substruct fields. These fields will be 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 this opcode to add the bearer information through the service extension to update the service flist.

GSM Manager FM Standard Opcodes

The opcode listed in Table 1–42 performs telco provisioning functions.

Header File

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

Opcode Index

Table 1–42 GSM Manager FM Standard Opcode

Opcode	Description	Use
PCM_OP_GSM_APPLY_PARAMETER	Updates the objects impacted by the product provisioning update.	Recommended

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.

This opcodes calls the PCM_OP_GSM_POL_APPLY_PARAMETER policy opcode to apply any customizations.

IC FM Standard Opcodes

The opcodes listed in Table 1–43 handle settlement information for inter-network operator usage.

Header File

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

Opcode Index

Table 1–43 IC FM Standard Opcodes

Opcode	Description	Use
PCM_OP_IC_DAILY_LOADER	Loads roaming settlement information into the BRM database. See the discussion on opcodes used for managing settlement data in *BRM Configuring Roaming in Pipeline Manager*.	Limited
PCM_OP_IC_LOAD_SMS_REPORT	Validates SMS aggregation data and creates or updates the SMS settlement report object. See the discussion on loading SMS data into the BRM database in *BRM Configuring Roaming in Pipeline Manager*.	Limited

PCM_OP_IC_DAILY_LOADER

Loads prerated roaming settlement information into the BRM database.

See the discussion on opcodes used for managing settlement data in *BRM Configuring Roaming in Pipeline Manager*.

PCM_OP_IC_LOAD_SMS_REPORT

Validates the SMS Aggregation data and creates or updates the **/sms_settle_report** object.

See the discussion on loading SMS data into the BRM database in *BRM Configuring Roaming in Pipeline Manager*.

IMT Manager FM Policy Opcodes

The opcodes in Table 1–44 are used to update the IMT and PDC objects impacted by the product provisioning update.

Header File

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

Opcode Index

Table 1–44 IMT Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_IMT_POL_APPLY_PARAMETER	Updates the IMT objects impacted by the product provisioning update.	Recommended
PCM_OP_PDC_POL_APPLY_PARAMETER	Updates the PDC objects impacted by the product provisioning update.	Recommended

PCM_OP_IMT_POL_APPLY_PARAMETER

Updates the IMT service object that was impacted by the product provisioning update.

When you add a new IMT service, you must customize this policy opcode to store attributes specific to that service in the BRM database. The attributes must match the product provisioning tags in **/config/telco/imt**.

Return Value

This opcode returns a copy of the input flist with any updates to custom fields in the service.

PCM_OP_PDC_POL_APPLY_PARAMETER

Updates the PDC service object that was impacted by the product provisioning update.

When you add a new PDC service, you must customize this policy opcode to store attributes specific to that service in the BRM database. The attributes must match the product provisioning tags in **/config/telco/pdc**.

Return Value

This opcode returns a copy of the input flist with any updates to custom fields in the service.

Invoicing FM Policy Opcodes

The opcodes listed in Table 1–45 are used to generate invoices in different formats.

Header File

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

Opcode Index

Table 1–45 Invoicing FM Policy Opcodes

Opcode	Description	Use
PCM_OP_INV_POL_FORMAT_INVOICE	Formats the invoice for printing. See the discussion on customizing the format for printed invoices in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_INV_POL_FORMAT_INVOICE_DOC1	Provides DOC1 formatted invoice. See the discussion on customizing the format for DOC1 invoices in *BRM Configuring and Running Billing*.	Limited
PCM_OP_INV_POL_FORMAT_INVOICE_HTML	Provides HTML formatted invoice. See the discussion on customizing the format for HTML invoices in *BRM Configuring and Running Billing*.	Limited
PCM_OP_INV_POL_FORMAT_INVOICE_XSLT	Provides XSL style sheet formatted invoice. See the discussion on customizing the invoice format by using an XSL style sheet in *BRM Configuring and Running Billing*.	Limited
PCM_OP_INV_POL_FORMAT_INVOICE_XML	Provides XML formatted invoice. See the discussion on customizing the format for XML invoices in *BRM Configuring and Running Billing*.	Limited
PCM_OP_INV_POL_FORMAT_VIEW_INVOICE	Formats invoices for viewing. See the discussion on displaying an invoice on demand in *BRM Configuring and Running Billing*.	Recommended

Table 1–45 (Cont.) Invoicing FM Policy Opcodes

Opcode	Description	Use
PCM_OP_INV_POL_POST_MAKE_INVOICE	Returns any errors encountered by PCM_OP_INV_POL_SELECT when performing custom invoicing.	Recommended
PCM_OP_INV_POL_PREP_INVOICE	Prepares the invoice. See the discussion on customizing invoice information in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_INV_POL_SELECT	Provides custom search templates for items and events. See the discussion on customizing invoice search operations in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_INV_POL_FORMAT_INVOICE

Specifies the format in which to store invoices.

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 **pin_flist**.

This opcode is called by the PCM_OP_INV_MAKE_INVOICE opcode.

See the discussion on customizing the format for printed invoices in *BRM Configuring and Running Billing*.

Example 1–187 Sample Input Flist

The input flist contains all the fields that you want to be included in the invoice.

Example 1–188 Sample Output Flist

This flist is returned when the default **pin_flist** storage format is specified:

```
0 PIN_FLD_FORMATS     ARRAY[*]
1 PIN_FLD_TYPE_STR     STR[0] "text/xml"
```

PCM_OP_INV_POL_FORMAT_INVOICE_DOC1

Provides the real formatted invoice for the DOC1 format.

If your system has the invoicing by service feature enabled, this opcode displays the invoice items by service instance.

This opcode is called by the PCM_OP_INV_POL_FORMAT_VIEW_INVOICE policy opcode.

See the discussion on customizing the format for DOC1 invoices in *BRM Configuring and Running Billing*.

PCM_OP_INV_POL_FORMAT_INVOICE_HTML

Provides the real formatted invoice for the HTML format.

This opcode is called by PCM_OP_INV_POL_FORMAT_VIEW_INVOICE when the invoice format requested is for HTML.

See the discussion on customizing the format for HTML invoices in *BRM Configuring and Running Billing*.

PCM_OP_INV_POL_FORMAT_INVOICE_XSLT

Provides the real formatted invoice with an XSL style sheet format.

This opcode is called by PCM_OP_INV_POL_FORMAT_VIEW_INVOICE when the **/config/invoice_templates** object specifies an XSL style sheet.

See the discussion on customizing the invoice format by using an XSL style sheet in *BRM Configuring and Running Billing*.

PCM_OP_INV_POL_FORMAT_INVOICE_XML

Provides the real formatted invoice for the XML format.

This opcode is called by PCM_OP_INV_POL_FORMAT_VIEW_INVOICE when the invoice format requested is for XML.

See the discussion on customizing the format for XML invoices *BRM Configuring and Running Billing*.

PCM_OP_INV_POL_FORMAT_VIEW_INVOICE

Generates an invoice on the fly in the specified format.

This opcode is called when the PCM_OP_INV_VIEW_INVOICE opcode requests an invoice in a format that is not stored on the **/invoice** object. This opcode attempts to generate the invoice in the requested format on the fly. Invoices may be formatted as HTML or DOC1. An XML format is also available, but it displays as HTML format.

See the discussion on displaying an invoice on demand in *BRM Configuring and Running Billing*.

PCM_OP_INV_POL_POST_MAKE_INVOICE

Captures and returns errors that were created due to custom invoicing operations defined in PCM_OP_INV_POL_SELECT.

This opcode is called by the PCM_OP_INV_MAKE_INVOICE standard opcode.

See the discussion on how invoices are generated in *BRM Configuring and Running Billing*.

Example 1–189 Sample Input Flist

```
0 PIN_FLD_POID          POID   [0] 0.0.0.1 /invoice -1 0
0 PIN_FLD_ERROR_CODE     STR   [0] "PIN_ERR_BAD_OPCODE"
0 PIN_FLD_ERROR_DESCR     STR   [0] "\t     <location=PIN_ERRLOC_FM:5 class=PIN_
ERRCLASS_SYSTEM_DETERMINATE:1 errno=PIN_ERR_BAD_OPCODE:36>\n\t
     <field num=0:0,0 recid=0 reserved=973 reserved2=0 time(sec:usec)=0:0>\n\t
     <facility=0 msg_id=0 version=0>\n"
```

Example 1–190 Sample Output Flist

```
0 PIN_FLD_POID           POID   [0] 0.0.0.1 /invoice -1 0
0 PIN_FLD_ERROR_CODE     STR    [0] "PIN_ERR_BAD_OPCODE"
0 PIN_FLD_ERROR_DESCR     STR    [0] "\t     <location=PIN_ERRLOC_FM:5 class=PIN_
ERRCLASS_SYSTEM_DETERMINATE:1 errno=PIN_ERR_BAD_OPCODE:36>\n\t
     <field num=0:0,0 recid=0 reserved=973 reserved2=0 time(sec:usec)=0:0>\n\t
     <facility=0 msg_id=0 version=0>\n"
```

PCM_OP_INV_POL_PREP_INVOICE

Prepares invoice information prior to formatting and storing.

This policy opcode is called by the PCM_OP_INV_MAKE_INVOICE opcode.

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 values. However, both Replacement and Correction Letter invoices show the previous amount, adjusted amount, and new amounts for events and items.

See the discussion on customizing invoice information in *BRM Configuring and Running Billing*.

When called by the PCM_OP_INV_MAKE_INVOICE opcode during trial invoicing for hierarchical accounts, this policy opcode checks the value of the PIN_FLD_FLAGS field. If the value is PIN_INV_TYPE_TRIAL_INVOICE plus PIN_INV_TYPE_SUBORDINATE this policy opcode adds the PIN_FLD_BILLS array and the PIN_FLD_CHECK_SPLIT_FLAGS field, which are required to generate the trial invoices for the subordinate'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 Business Intelligence (BI) Publisher to collate subordinate invoices into the final invoice.

For more information, see the discussion on how trial invoicing works in *BRM Designing and Generating Invoices*.

PCM_OP_INV_POL_SELECT

Provides the ability to write custom search templates for items and events to be displayed on invoices.

The PIN_FLD_BOOLEAN field in the output flist determines whether this opcode is used:

- **PIN_BOOLEAN_TRUE** (**1**) means the output flist is used by PCM_OP_INV_POL_POST_MAKE_INVOICE to generate the invoice.
- **PIN_BOOLEAN_FALSE** (**0**) means it is not used.

This opcode is called by the PCM_OP_INV_MAKE_INVOICE standard opcode.

See the discussion on customizing invoice search operations in *BRM Configuring and Running Billing*.

Invoicing FM Standard Opcodes

The opcodes in Table 1–46 create and format invoices.

Header File

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

Opcode Index

Table 1–46 Invoicing FM Standard Opcodes

Opcode	Description	Use
PCM_OP_INV_DECODE_INVOICE_DATA	Decodes the value of the PIN_FLD_INVOICE_DATA field.	Recommended
PCM_OP_INV_FORMAT_INVOICE	Performs XSL Transformation on an invoice. See the discussion on how invoices are formatted in *BRM Configuring and Running Billing*.	Limited
PCM_OP_INV_MAKE_INVOICE	Generates invoices. See the discussion on how invoices are generated in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_INV_VIEW_INVOICE	Displays an invoice that is stored in the database. See the discussion on customizing the format for online invoices in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_INV_DECODE_INVOICE_DATA

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.

If you customized PCM_OP_INV_POL_SELECT to search for custom event data, you must call this opcode.

See the discussion on decoding cached event data for invoicing in *BRM Configuring and Running Billing*.

PCM_OP_INV_FORMAT_INVOICE

Performs XSL Transformation on an invoice.

See the discussion on how invoices are formatted in *BRM Configuring and Running Billing*.

PCM_OP_INV_MAKE_INVOICE

Creates an invoice for a specified (regular or corrective) bill object.

For regular invoices, this opcode uses the PIN_FLD_INV_DETAIL_FLAG value in the **/payinfo** object to determine whether to generate a detailed invoice or a summary invoice, and the invoicing threshold parameters in the **/config/business_params** object to determine whether the invoices of subordinate bills in a hierarchy should be generated separately or consolidated into the invoice of the parent A/R bill.

This is the initial opcode that gets called to create an invoice for a designated bill object.

This opcode 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 an 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 this opcode is called, the opcode discards the default value and uses the PIN_FLD_INV_FLAGS value from the input flist in conjunction with the entries in the **pin_inv.h** file to arrive at the required type of corrective invoice.

This 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.

See the discussion on how invoices are generated in *BRM Configuring and Running Billing*.

During trial invoicing for hierarchical accounts, the PCM_OP_BILL_MAKE_TRIAL_BILL opcode passes the PIN_FLD_INV_FLAGS field's value to this opcode. This opcode decides from the value whether to create a subordinate or parent trial invoice:

- If the value is PIN_INV_TYPE_SUBORDINATE, a subordinate invoice is created.
- If the value is PIN_INV_TYPE_PARENT, a parent invoice is created.

For more information, see the discussion on how trial invoicing works in *BRM Designing and Generating Invoices*.

Error codes

PIN_ERR_NO_MEM

Insufficient memory to complete the operation.

PIN_ERR_BAD_ARG

A required field in an flist is incorrect.

PCM_OP_INV_VIEW_INVOICE

Retrieves an invoice from the database.

This opcode uses the POID of the **/bill** object or **/invoice** object to locate and retrieve a specific invoice and the PIN_FLD_THRESHOLD value to determine the maximum allowable size of the invoice to be viewed.

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 account, 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.

See the discussion on customizing the format for online invoices in *BRM Configuring and Running Billing*.

IP Address Manager APN FM Policy Opcodes

The opcodes in Table 1–47 are used to perform various checks and maintenance on the APN device.

Header File

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

Opcode Index

Table 1–47 IP Address Manager APN FM Policy Opcodes

Opcode	Description	Use
PCM_OP_APN_POL_DEVICE_ASSOCIATE	Performs verification checks on the APN device and the account or service being associated. See the discussion on associating APN with an account or service in *BRM Telco Integration*.	Recommended
PCM_OP_APN_POL_DEVICE_CREATE	Performs validation checks during APN device creation. See the discussion on creating an APN device in *BRM Telco Integration*.	Recommended
PCM_OP_APN_POL_DEVICE_DELETE	Performs validation checks and deletes associated IP devices during APN device deletion. See the discussion on deleting an APN device in *BRM Telco Integration*.	Recommended
PCM_OP_APN_POL_DEVICE_SET_ATTR	Verifies that an APN device is in a state to accept changes, and then makes the changes. See the discussion on modifying an APN device in *BRM Telco Integration*.	Recommended
PCM_OP_APN_POL_DEVICE_SET_BRAND	Verifies that an APN device can accept a new or changed brand, and then makes the change to the APN and all associated IP devices. See the discussion on setting the brand for an APN device in *BRM Telco Integration*.	Recommended
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. See the discussion on the discussion on changing the APN device state in *BRM Telco Integration*.	Recommended

PCM_OP_APN_POL_DEVICE_ASSOCIATE

Performs verification checks on the APN device and the account or service being associated.

This opcode is called by the PCM_OP_DEVICE_POL_ASSOCIATE policy opcode.

See the discussion on associating APN with an account or service in *BRM Telco Integration*.

By default, this opcode first verifies the following:

- The APN device is in a **new** or **usable** state.
- The account or service being associated are in the same brand as the APN device.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–191 Sample Input Flist

```
0  PIN_FLD_POID             POID [0] 0.0.0.1 /device/apn 109867 0
0  PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0  PIN_FLD_FLAGS            INT [0] 1
0  PIN_FLD_SERVICES         ARRAY [0]
1    PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 113976 0
1    PIN_FLD_SERVICE_OBJ   POID [0] 0.0.0.1 /service/ip 112056 8
```

PCM_OP_APN_POL_DEVICE_CREATE

Performs validation checks during APN device creation. This opcode verifies that the APN names and IDs being created are correct and determines whether there are any existing APN devices with duplicate names.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

This opcode is called by the PCM_OP_DEVICE_POL_CREATE policy opcode.

See the discussion on creating an APN device in *BRM Telco Integration*.

Example 1–192 Sample Input Flist

```
0  PIN_FLD_POID         POID [0] 0.0.0.1 /device/apn 0 -1
0  PIN_FLD_PROGRAM_NAME STR [0] "testnap"
0  PIN_FLD_DEVICE_ID    STR [0] "TESTINGAPN"
0  PIN_FLD_STATE_ID     INT [0] 1
0  PIN_FLD_DESCR        STR [0] "Sample Device"
```

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.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

This opcode is called by the PCM_OP_DEVICE_POL_DELETE policy opcode.

See the discussion on deleting an APN device in *BRM Telco Integration*.

Example 1–193 Sample Input Flist

```
0  PIN_FLD_POID          POID [0] 0.0.0.1 /device/apn 857148 0
0  PIN_FLD_PROGRAM_NAME   STR [0] "testnap"
```

PCM_OP_APN_POL_DEVICE_SET_ATTR

Verifies that an APN device is in a state to accept changes, and then makes the changes.

By default, this opcode first confirms the following:

- The APN device is associated with the correct brand.
- The APN device object type is **/device/apn**.
- None of the IP addresses associated with the APN device are in an **allocated** state.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

This opcode is called by the PCM_OP_DEVICE_POL_SET_ATTR policy opcode.

See the discussion on modifying an APN device in *BRM Telco Integration*.

Example 1–194 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /device/apn 62756 1
0 PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0 PIN_FLD_DESCR         STR [0] "change apn"
0 PIN_FLD_DEVICE_ID     STR [0] "PORTAL"
```

PCM_OP_APN_POL_DEVICE_SET_BRAND

Verifies that an APN device can accept a new or changed brand, and then makes the change to the APN and all associated IP devices.

This opcode is called by the PCM_OP_DEVICE_POL_SET_BRAND and PCM_OP_IP_POL_DEVICE_SET_BRAND policy opcodes.

See the discussion on setting the brand for an APN device in *BRM Telco Integration*.

This opcode first confirms the following:

- The device type is **/device/apn**.
- The APN device state is **new** or **unusable**.
- None of the IP address devices associated with the APN are in an **allocated** state.
- That PCM_OP_IP_POL_DEVICE_CREATE is attempting to change the APN device state from **new** to **usable**.

If these conditions are met, this opcode calls PCM_OP_DEVICE_SET_BRAND to add or change the APN device's brand. All IP address devices associated with the APN device are also given the new brand.

If any of these conditions fail, no changes is made to the APN or IP address devices.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–195 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /device/apn 62756 1
0 PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 113976 0
```

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.

This opcode is called by the PCM_OP_DEVICE_POL_SET_STATE policy opcode.

See the discussion on changing the APN device state in *BRM Telco Integration*.

By default, this opcode first 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.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–196 Sample Input Flist

```
0  PIN_FLD_POID          POID [0] 0.0.0.1 /device/apn 698934 1
0  PIN_FLD_PROGRAM_NAME  STR [0] "testnap"
0  PIN_FLD_DEVICE_ID     STR [0] "testing_apn"
0  PIN_FLD_NEW_STATE      INT [0] 2
```

IP Address Manager FM Policy Opcodes

The opcodes in Table 1–48 are used to perform various checks and maintain the state of the IP address device.

Header File

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

Opcode Index

Table 1–48 IP Address Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_IP_POL_DEVICE_ASSOCIATE	Changes the state of an IP address device when services are assigned to, or removed from the device. See the following discussions in *BRM Telco Integration*: ■ Associating an IP address with accounts or services ■ Disassociating an IP address device from accounts or services	Recommended
PCM_OP_IP_POL_DEVICE_CREATE	Executes verification checks before creating a **/device/ip** object. See the following discussions in *BRM Telco Integration*: ■ Creating a single IP address device ■ Creating a range of IP address devices	Recommended
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. See the discussion on deleting an IP address device in *BRM Telco Integration*.	Recommended
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 of the **/device/ip** object. See the discussion on modifying an IP address device in *BRM Telco Integration*.	Recommended
PCM_OP_IP_POL_DEVICE_SET_BRAND	Verifies that an IP address device is eligible to change brands, and then makes the change. See the discussion on setting the brand on an IP device in *BRM Telco Integration*.	Recommended
PCM_OP_IP_POL_DEVICE_SET_STATE	Changes the state of all instances of an IP device. See the discussion on changing IP device states from unallocated to returned in *BRM Telco Integration*.	Recommended

PCM_OP_IP_POL_DEVICE_ASSOCIATE

Changes the state of an IP address device when services are assigned to, or removed from the device.

- When a service is assigned to an IP device, this opcode changes the state of the IP device from **new** or **unallocated** to **allocated**.
- When all services are removed from an IP device, this opcode changes the state of the IP device from **allocated** to **unallocated**.

This opcode is called by the PCM_OP_DEVICE_POL_ASSOCIATE and PCM_OP_IP_POL_DEVICE_SET_STATE policy opcodes.

See the following discussions in *BRM Telco Integration*:

- Associating an IP address with accounts or services
- Disassociating an IP address device from accounts or services

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–197 Sample Input Flist

```
0  PIN_FLD_POID             POID [0] 0.0.0.1 /device/ip 10986 0
0  PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0  PIN_FLD_FLAGS            INT [0] 1
0  PIN_FLD_SERVICES         ARRAY [0]
1    PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 113976 0
1    PIN_FLD_SERVICE_OBJ   POID [0] 0.0.0.1 /service/ip 112056 8
```

Example 1–198 Sample Output Flist

The input flist is returned.

PCM_OP_IP_POL_DEVICE_CREATE

Executes verification checks before creating a **/device/ip** object.

This opcode verifies that no existing IP address devices have the same IP address and APN combination.

This opcode is called by the PCM_OP_DEVICE_POL_CREATE and PCM_OP_IP_POL_DEVICE_SET_STATE policy opcodes.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

See the following discussions in *BRM Telco Integration*:

- Creating a single IP address device
- Creating a range of IP address devices

Example 1–199 Sample Input Flist

```
0  PIN_FLD_POID             POID [0] 0.0.0.1 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME     STR  [0] "testnap"
0  PIN_FLD_DEVICE_ID        STR  [0] "207.1.0.100"
0  PIN_FLD_DEVICE_IP  SUBSTRUCT  [0]
1      PIN_FLD_APN_OBJ       POID [0] 0.0.0.1 /device/apn 68728  0
```

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.

This opcode is designed as a hook for you to add any additional verification checks or business that your implementation requires.

This opcode is called by the PCM_OP_DEVICE_POL_DELETE policy opcode.

See the discussion on deleting an IP address device in *BRM Telco Integration*.

Example 1–200 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 62756 1
```

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.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

This opcode is called by the PCM_OP_DEVICE_POL_SET_ATTR and PCM_OP_IP_POL_DEVICE_SET_STATE policy opcodes.

See the discussion on modifying an IP address device in *BRM Telco Integration*.

Example 1–201 Sample Input Flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 71115 1
0 PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0 PIN_FLD_DESCR         STR [0] "change apn"
```

PCM_OP_IP_POL_DEVICE_SET_BRAND

Verifies that an IP address device is eligible to change brands, and then makes the change.

This opcode is called by the PCM_OP_DEVICE_POL_SET_BRAND policy opcode.

See the discussion on setting the brand on an IP device in *BRM Telco Integration*.

By default, this opcode first verifies the following:

- The calling opcode is PCM_OP_IP_POL_DEVICE_SET_BRAND.
- The IP device is not in a **new** or **unallocated** state.

This opcode also checks whether the brand passed in matches the brand the IP address already has. If so, it returns a debug message.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–202 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /device/ip 62756 1
0 PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0 PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account 113976 0
```

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.

This opcode is called by the PCM_OP_DEVICE_POL_SET_STATE policy opcode.

See the discussion on changing IP device states from unallocated to returned in *BRM Telco Integration*.

This opcode 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.

This opcode is designed as a hook for you to add any additional verification checks or business logic that your implementation requires.

Example 1–203 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /device/ip 692807
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_NEW_STATE        INT [0] 3
```

IP Address Manager FM Standard Opcodes

The opcodes in Table 1–49 create, delete, and maintain the attributes and state of one or more IP address devices.

Header File

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

Opcode Index

Table 1–49 IP Address Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_IP_DEVICE_CREATE	Creates one or more IP address devices. See the discussion on creating a single IP address device and creating a range of IP address devices in *BRM Telco Integration*.	Recommended
PCM_OP_IP_DEVICE_DELETE	Deletes one or more IP address devices. See the discussion on deleting an IP address device in *BRM Telco Integration*.	Recommended
PCM_OP_IP_DEVICE_SET_ATTR	Sets attributes for one or more IP devices. See the discussion on modifying an IP address device in *BRM Telco Integration*.	Recommended
PCM_OP_IP_DEVICE_SET_STATE	Sets the device state for one or more IP devices. See the discussion on changing IP device states from unallocated to returned in *BRM Telco Integration*.	Recommended

PCM_OP_IP_DEVICE_CREATE

Creates one or more IP address devices. This opcode calls PCM_OP_DEVICE_CREATE. Add any validation checks or business logic that your business requires before creating and IP device to PCM_OP_IP_POL_DEVICE_CREATE.

See the discussion on creating a single IP address device and creating a range of IP address devices in *BRM Telco Integration*.

When creating multiple IP devices, this opcode creates a contiguous range of IP addresses devices based on starting and ending IP addresses you supply. The opcode creates a **/device/ip** object for each device.

You have the option of using wildcard characters in the last two octets of the PIN_FLD_START_ADDRESS field. If they are used, this opcode creates a range using them.

If wildcard characters are not used, this opcode performs one of the following actions:

- Creates a range using the end address
- Creates a range using subnet mask
- Creates a single IP device

Example 1–204 Sample Input Flists

All of these sample input flists work.

```
0  PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip -1 0
0  PIN_FLD_OBJ_TYPE     STR [0] "/ext"
0  PIN_FLD_PROGRAM_NAME STR  [0] "testnap"
0  PIN_FLD_START_ADDRESS   STR  [0] "207.1.2.10"
0  PIN_FLD_END_ADDRESS     STR  [0] "207.1.2.19"
0  PIN_FLD_APN_OBJ         POID [0] 0.0.0.1 /device/apn 68728  0
0  PIN_FLD_EXTENDED_INFO   SUBSTRUCT [0]
1      PIN_FLD_ORDER_INFO      SUBSTRUCT [0]
2         PIN_FLD_ORDER_ORIGIN     STR [0] "northcoast"
```

Or

```
0  PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME STR  [0] "testnap"
0  PIN_FLD_START_ADDRESS   STR  [0] "207.1.2.10"
0  PIN_FLD_APN_OBJ         POID [0] 0.0.0.1 /device/apn 68728  0
```

Or

```
0  PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME STR  [0] "testnap"
0  PIN_FLD_START_ADDRESS   STR  [0] "207.1.2.10"
0  PIN_FLD_SUBNET_MASK     STR  [0] "255.255.255.192"
0  PIN_FLD_APN_OBJ         POID [0] 0.0.0.1 /device/apn 68728  0
```

Or

```
0  PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME STR  [0] "testnap"
0  PIN_FLD_START_ADDRESS   STR  [0] "207.1.*.*"
0  PIN_FLD_APN_OBJ         POID [0] 0.0.0.1 /device/apn 68728  0
```

Example 1–205 Sample Output Flist

```
0  PIN_FLD_POID        POID [0] 0.0.0.1 /device/ip -1 0
```

```
0  PIN_FLD_RESULTS    ARRAY [0]
1    PIN_FLD_POID      POID [0] 0.0.0.1 /device/ip 2723 0
0  PIN_FLD_RESULTS    ARRAY [1]
1    PIN_FLD_POID      POID [0] 0.0.0.1 /device/ip 5654 0
0  PIN_FLD_RESULTS    ARRAY [2]
1    PIN_FLD_POID      POID [0] 0.0.0.1 /device/ip 7653 0
0  PIN_FLD_RESULTS    ARRAY [3]
1    PIN_FLD_POID      POID [0] 0.0.0.1 /device/ip 90283 0
0  PIN_FLD_RESULTS    ARRAY [4]
1    PIN_FLD_POID       POID [0] 0.0.0.1 /device/ip 3754 0
```

Return Values

This opcode returns an array containing the POID of each IP device created.

PCM_OP_IP_DEVICE_DELETE

Deletes one or more IP address devices. 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.

For more information and the calling sequence, see the discussion on deleting an IP address device in *BRM Telco Integration*.

Example 1–206 Sample Input Flists

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /device/ip -1 0(Routing POID)
0  PIN_FLD_PROGRAM_NAME   STR [0] "Testnap"
0  PIN_FLD_ARGS           ARRAY [0]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 742181 1
0  PIN_FLD_ARGS           ARRAY [1]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 742949 1
0  PIN_FLD_ARGS           ARRAY [2]
1    PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 743717 1
```

Example 1–207 Sample Output Flist

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /device/ip 1 0
0  PIN_FLD_RESULTS        ARRAY [0] allocated 1, used 1
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 742181 0
0  PIN_FLD_RESULTS        ARRAY [1] allocated 1, used 1
1    PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 742949 0
```

PCM_OP_IP_DEVICE_SET_ATTR

Sets the attributes for one or more IP address devices. This opcode calls PCM_OP_DEVICE_SET_ATTR. Add any validation checks or business logic before setting IP device attributes to PCM_OP_IP_POL_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.

See the discussion on modifying an IP address device in *BRM Telco Integration*.

Example 1–208 Sample Input Flist

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 0 0 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME   STR [0] "Testnap"
0  PIN_FLD_ARGS           ARRAY [0]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 743717 1
1    PIN_FLD_DESCR        STR [0] "North"
1    PIN_FLD_DEVICE_IP    PIN_FLDT_SUBSTRUCT [0]
2      PIN_FLD_APN_OBJ    POID [0] 0.0.0.1 /device/apn 732512 1

0  PIN_FLD_ARGS           ARRAY [1]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 743718 1
1    PIN_FLD_DEVICE_IP    PIN_FLDT_SUBSTRUCT [0]
2    PIN_FLD_APN_OBJ      POID [0] 0.0.0.1 /device/apn 732512 1

0  PIN_FLD_ARGS           ARRAY [2]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 745253 1
1    PIN_FLD_DESCR         STR [0] "North"
```

Example 1–209 Sample Output Flist

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /device/ip 1 0
0  PIN_FLD_RESULTS        ARRAY [0] allocated 1, used 1
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 742181 0
0  PIN_FLD_RESULTS        ARRAY [1] allocated 1, used 1
1    PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 742949 0
```

PCM_OP_IP_DEVICE_SET_STATE

Sets the device state for one or more IP devices. This opcode calls PCM_OP_DEVICE_SET_STATE. Add any validation checks or business logic required before setting device states to PCM_OP_IP_POL_DEVICE_SET_STATE.

When setting states for multiple IP devices, this opcode 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.

See the discussion on changing IP device states from unallocated to returned in *BRM Telco Integration*.

Example 1–210 Sample Input Flists

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 0 0 /device/ip -1 0
0  PIN_FLD_PROGRAM_NAME   STR [0] "Testnap"
0  PIN_FLD_ARGS           ARRAY [0]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 766144 1
1    PIN_FLD_NEW_STATE    INT [0] 4

0  PIN_FLD_ARGS           ARRAY [1]
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 766145 1
1    PIN_FLD_NEW_STATE     INT [0] 4
```

Example 1–211 Sample Output Flist

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /device/ip 1 0
0  PIN_FLD_RESULTS        ARRAY [0] allocated 1, used 1
1    PIN_FLD_POID         POID [0] 0.0.0.1 /device/ip 742181 0
0  PIN_FLD_RESULTS        ARRAY [1] allocated 1, used 1
1    PIN_FLD_POID          POID [0] 0.0.0.1 /device/ip 742949 0
```

Job FM Standard Opcodes

The opcodes listed in Table 1–50 are the standard opcodes to create, modify, or delete the **/job_template** objects.

Header File

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

Opcode Index

Table 1–50 Job FM Standard Opcodes

Opcode	Description	Use
PCM_OP_JOB_PROCESS_TEMPLATE	Creates, modifies, or deletes the **/job_template** objects.	Recommended

PCM_OP_JOB_PROCESS_TEMPLATE

Creates, modifies, or deletes the **/job_template** objects created through Oracle Communications Billing and Revenue Management Business Operations Center.

LDAP Base Opcodes

The opcodes listed in Table 1–51 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.

> **Note:** Each of the DMs included with BRM uses a different implementation of the base opcodes depending on the DM and the storage system it interacts with. For example, the base opcode PCM_OP_SEARCH is implemented differently for the DM_ORACLE and the DM_LDAP.

Header File

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

Opcode Index

Table 1–51 LDAP Base Opcodes

Opcode	Description	Use
PCM_OP_CREATE_OBJ	Creates an entry in the directory server. See the discussion on creating directory server entries in *BRM LDAP Manager*.	Recommended
PCM_OP_DELETE_FLDS	Deletes values and attributes in an entry using the LDAP modify operation. See the discussion on deleting attributes from an existing directory server entry in *BRM LDAP Manager*.	Recommended
PCM_OP_DELETE_OBJ	Invokes the delete entry semantics of the LDAP modify operation in the directory server. See the discussion on deleting directory server entries in *BRM LDAP Manager*.	Recommended
PCM_OP_READ_FLDS	Reads attributes from a directory server entry from the database using the LDAP search operation. See the discussion on reading attributes from the directory server entry in *BRM LDAP Manager*.	Recommended
PCM_OP_READ_OBJ	Reads an entire object from the database using the LDAP search operation. See the discussion on reading objects from the directory server in *BRM LDAP Manager*.	Recommended

Table 1–51 (Cont.) LDAP Base Opcodes

Opcode	Description	Use
PCM_OP_SEARCH	Searches the directory server based on a specified search criteria. See the discussion on searching the directory server for entries in *BRM LDAP Manager*.	Recommended
PCM_OP_TEST_LOOPBACK	Tests directory server connections.	Recommended
PCM_OP_WRITE_FLDS	Updates attributes in an entry or renames entries using the LDAP modify operation. See the discussion on changing directory server entries in *BRM LDAP Manager*.	Recommended

PCM_OP_CREATE_OBJ

Creates new directory server entries or reuses entries in the directory server for replication purposes.

See the discussion on creating directory server entries in *BRM LDAP Manager*.

PCM_OP_DELETE_FLDS

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.

See the discussion on deleting attributes from an existing directory server entry in *BRM LDAP Manager*.

PCM_OP_DELETE_OBJ

Deletes an entire entry from the directory server.

See the discussion on deleting directory server entries in *BRM LDAP Manager*.

PCM_OP_READ_FLDS

Reads attributes from a directory server.

See the discussion on reading attributes from the directory server entry in *BRM LDAP Manager*.

PCM_OP_READ_OBJ

Reads objects from a directory server entry using the LDAP *search* operation.

See the discussion on reading objects from the directory server in *BRM LDAP Manager*.

PCM_OP_SEARCH

Searches the directory server based on a specified search criteria that you supply as a template in the input flist.

Important: *Only* those objects and attributes that you define in the mapping file can be returned by the LDAP Data Manager in the output flist.

See the discussion on searching the directory server for entries in *BRM LDAP Manager*.

PCM_OP_TEST_LOOPBACK

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

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

See the discussion on changing directory server entries in *BRM LDAP Manager*.

Number Manager FM Policy Opcodes

Use the opcodes in Table 1–52 to customize Number Manager.

Header File

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

Opcode Index

Table 1–52 Number Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_NUM_POL_CANONICALIZE	Handles number normalization, for example, removes punctuation characters. See the discussion on customizing number normalization in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_POL_DEVICE_ASSOCIATE	Validates that numbers are associated with services correctly. See the discussion on customizing how numbers are associated with services in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_POL_DEVICE_CREATE	Validates the default telephone number attributes when a number is created. See the discussion on customizing telephone number attributes in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_POL_DEVICE_DELETE	Checks the state of the device.	
PCM_OP_NUM_POL_DEVICE_SET_ATTR	Specifies how a number can be changed, for example, which digits can be changed. See the discussion on customizing how a number can be changed in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_POL_DEVICE_SET_BRAND	Changes a block's brand, and the brand of all numbers in the block. See the discussion on changing a block's brand in *BRM Telco Integration*.	Recommended

PCM_OP_NUM_POL_CANONICALIZE

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

See the discussion on customizing number normalization in *BRM Telco Integration*.

Example 1–212 Sample Input Flists

- This example shows a typical input format:

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 0 0
0 PIN_FLD_NUMBERS          ARRAY [0]
1 PIN_FLD_NUMBER            STR [0] "1 (408) 572-3333"
```

- This example shows how multiple numbers are handled:

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 0 0
0 PIN_FLD_NUMBERS          ARRAY [0]
1 PIN_FLD_NUMBER      STR [0] "1 (408) 572-3000"
0 PIN_FLD_NUMBERS          ARRAY [1]
1 PIN_FLD_NUMBER        STR [0] "1 (408) 572-3999"
```

Example 1–213 Sample Output Flist

This example shows the default output:

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 0 0
0 PIN_FLD_NUMBERS          ARRAY [0]
1 PIN_FLD_NUMBER            STR [0] "0014085723333"
```

PCM_OP_NUM_POL_DEVICE_ASSOCIATE

Specifies the rules for associating or disassociate a number and a service.

This opcode is called by the PCM_OP_DEVICE_POL_ASSOCIATE opcode when a number is associated or disassociated with a service.

See the discussion on customizing how numbers are associated with services in *BRM Telco Integration*.

PCM_OP_NUM_POL_DEVICE_CREATE

Validates a new number to make sure it is unique in the database.

You can customize this policy opcode if you extend the number device attributes and require additional validation, or if you want to change existing validations.

This opcode is called by the PCM_OP_DEVICE_POL_CREATE policy opcode.

See the discussion on customizing telephone number attributes in *BRM Telco Integration*.

PCM_OP_NUM_POL_DEVICE_DELETE

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 policy opcode is called by the PCM_OP_DEVICE_POL_DELETE policy opcode.

PCM_OP_NUM_POL_DEVICE_SET_ATTR

Specifies which digits in a number can be changed.

By default, it supports changing US area codes by using the following logic: If the number starts with 001 and is 13 digits long, allow changing digits 4-6.

This opcode is called by the PCM_OP_DEVICE_POL_SET_ATTR policy opcode.

See the discussion on customizing how a number can be changed in *BRM Telco Integration*.

Example 1–214 Sample input and output flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /device/num 15822
0 PIN_FLD_DEVICE_ID       STR  [0] "1-888-7772900"
```

PCM_OP_NUM_POL_DEVICE_SET_BRAND

Changes a block's brand, and the brand of all numbers in the block.

This opcode is called by the PCM_OP_NUM_MODIFY_BLOCK standard opcode.

See the discussion on changing a block's brand in *BRM Telco Integration*.

Example 1–215 Sample input and output flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /device/num 13768
0 PIN_FLD_ACCOUNT_OBJ        POID [0] 0.0.0.1 /account 234
```

Number Manager FM Standard Opcodes

The opcodes listed in Table 1–53 are used to create and modify blocks of numbers, manage number quarantine, and manage number portability.

Header File

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

Opcode Index

Table 1–53 Number Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_NUM_CREATE_BLOCK	Creates a block of telephone numbers. See the discussion on creating blocks of numbers in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_MODIFY_BLOCK	Modifies a block of numbers, for example, changes the block name, brand, numbers, or splits the block. See the discussion on modifying blocks of numbers in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_PORT_IN	Creates a number device with the provided number. See the discussion on managing number portability in *BRM Telco Integration*.	Limited
PCM_OP_NUM_PORT_OUT	Sets the status of the provided number device to **quarantine_port_out**. See the discussion on managing number portability in *BRM Telco Integration*.	Limited
PCM_OP_NUM_QUARANTINE	Either creates or deletes a **/schedule/device** object to manage the telephone number quarantine. See the discussion on managing number quarantine in *BRM Telco Integration*.	Recommended
PCM_OP_NUM_SPLIT_BLOCK	Splits an existing block of numbers into two or more blocks. See the discussion on splitting blocks of numbers in *BRM Telco Integration*.	Recommended

PCM_OP_NUM_CREATE_BLOCK

Creates a block of telephone numbers. This opcode creates a **/block** object and the specified number of telephone numbers, created as **/device/num** objects.

See the discussion on creating blocks of numbers in *BRM Telco Integration*.

Example 1–216 Sample Input Flist

```
0 PIN_FLD_NAME              STR[0] "Cupertino"
0 PIN_FLD_START_NUMBER      STR[0] "1 (408) 572-0000"
0 PIN_FLD_END_NUMBER        STR[0] "1 (408) 572-9999"
0 PIN_FLD_PROGRAM_NAME      STR[0] "Number Administrator"
0 PIN_FLD_NUMBER_INFO       ARRAY[0]
1 PIN_FLD_CATEGORY_ID       INT[0] 2
1 PIN_FLD_CATEGORY_VERSION  INT[0] 1
1 PIN_FLD_NETWORK_ELEMENT   STR[0] "HLR1"
1 PIN_FLD_VANITY            ENUM[0] 0
1 PIN_FLD_PERMITTED         STR[0] "/service/telco/gsm/telephony"
```

Example 1–217 Sample Output Flist

```
0 PIN_FLD_POID  POID  [0] 0.0.0.1  /block  12031 0
```

PCM_OP_NUM_MODIFY_BLOCK

Modifies a block of numbers, for example, changes the block name, brand, or numbers.

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**, the PCM_OP_NUM_MODIFY_BLOCK opcode extends or shrinks the number block. If the value of the flag is **False**, the PCM_OP_NUM_MODIFY_BLOCK opcode modifies the number block.

See the discussion on modifying blocks of numbers in *BRM Telco Integration*.

Example 1–218 Sample Input Flist

This example shows a block name change:

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /block 12301 0
0 PIN_FLD_PROGRAM_NAME    STR  [0] "Number Administrator"
0 PIN_FLD_NAME             STR  [0] "Northern California"
```

Example 1–219 Sample Output Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /block 12301 0
```

PCM_OP_NUM_PORT_IN

Creates a number device using the provided number.

See the discussion on managing number portability in *BRM Telco Integration*.

Example 1–220 Sample Input Flist

```
0 PIN_FLD_NAME                  STR  [0] "PORT_IN_123"
0 PIN_FLD_PORT_IN_NUMBER        STR  [0] "14085723700"
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /block -1 0
0 PIN_FLD_NUMBER_INFO           ARRAY[0]
1 PIN_FLD_PERMITTED             STR  [0] "/service/gsm"
1 PIN_FLD_VANITY                ENUM [0] 0
1 PIN_FLD_CATEGORY_ID           INT  [0] 0
1 PIN_FLD_NETWORK_ELEMENT       STR  [0] "sample_network_element_1"
1 PIN_FLD_ORIGIN_NETWORK_ID     STR  [0] "Donor Service Provider"
1 PIN_FLD_RECENT_NETWORK_ID     STR  [0] "sample_network_element_2"
1 PIN_FLD_CATEGORY_VERSION       INT  [0] 0
```

Example 1–221 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /device/num 15084 0
```

PCM_OP_NUM_PORT_OUT

Sets the status of the provided number device to **quarantine_port_out**.

See the discussion on managing number portability in *BRM Telco Integration*.

Example 1–222 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /device/num 15084 0
0 PIN_FLD_OLD_STATE      INT  [0] 2
0 PIN_FLD_NEW_STATE       INT  [0] 9
```

Example 1–223 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /device/num 15084 0
0 PIN_FLD_OLD_STATE      INT  [0] 2
0 PIN_FLD_NEW_STATE       INT  [0] 9
```

PCM_OP_NUM_QUARANTINE

Creates or deletes a **/schedule/device** object to manage the telephone number quarantine.

See the discussion on managing number quarantine in *BRM Telco Integration*.

Example 1–224 Sample input and output flist

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /device/num 18526
0 PIN_FLD_OLD_STATE     INT  [0] 2
0 PIN_FLD_NEW_STATE      INT  [0] 3
```

PCM_OP_NUM_SPLIT_BLOCK

Splits a block of numbers into two or more blocks.

See the discussion on splitting blocks of numbers in *BRM Telco Integration*.

Example 1–225 Sample Input Flist

This example shows a block split into two blocks:

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /block 15649
0 PIN_FLD_ATTRIBUTES      ARRAY [0]
1 PIN_FLD_NAME            STR [0] "Ohio, Summit County, Akron"
1 PIN_FLD_START_NUMBER    STR [0] "12167772000"
1 PIN_FLD_END_NUMBER      STR [0] "12167772499"
0 PIN_FLD_ATTRIBUTES      ARRAY [1]
1 IN_FLD_NAME             STR [0] "Ohio, Summit County, Akron"
1 PIN_FLD_START_NUMBER    STR [0] "12167772500"
1 PIN_FLD_END_NUMBER      STR [0] "12167772999"
```

Example 1–226 Sample Output Flist

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /block 15649 0
0 PIN_FLD_POIDS       ARRAY [0]
1 PIN_FLD_POID        POID [0] 0.0.0.1 /block 12031 0
0 PIN_FLD_POIDS       ARRAY [1]
1 PIN_FLD_POID         POID [0] 0.0.0.1 /block 45732 0
```

Order FM Policy Opcodes

Use the opcodes in Table 1–54 to customize order management.

Header File

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

Opcode Index

Table 1–54 Order FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ORDER_POL_ASSOCIATE	Can be customized to provide validation for associations and disassociations. See the discussion on associating and disassociating order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_CREATE	Can be customized to provide validation and other functionality during device creation. See the discussion on creating order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_DELETE	Can be customized to provide validation for device deletions. See the discussion on deleting order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_PROCESS	Can be customized to provide validation for processing order response files. See the discussion on processing order response files in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_SET_ATTR	Can be customized to provide validation for attribute changes. See the discussion on changing order object attributes in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_SET_BRAND	Can be customized to provide validation or other functionality during a brand change. See the discussion on changing order object brand associations in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_POL_SET_STATE	Can be customized to provide validation or other functionality for state changes. See the discussion on setting the state in order objects in *BRM Developer's Guide*.	Recommended

PCM_OP_ORDER_POL_ASSOCIATE

Allows custom validation during order association and disassociation.

This opcode is called by the PCM_OP_ORDER_ASSOCIATE standard opcode and takes the same input as the PCM_OP_ORDER_ASSOCIATE standard opcode.

See the discussion on associating and disassociating order objects in *BRM Developer's Guide*.

Example 1–227 Sample Input Flist

Associating an order with a master order:

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0  PIN_FLD_FLAGS             INT [0] 1
0  PIN_FLD_ORDERS            ARRAY [0]
1  PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.1 /service/ip 10433 0
1  PIN_FLD_ACCOUNT_OBJ        POID [0] 0.0.0.1 /account 11841 0
```

Disassociating an order from a master order:

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0  PIN_FLD_FLAGS             INT [0] 0
0  PIN_FLD_SERVICES          ARRAY [0]
1  PIN_FLD_SERVICE_OBJ        POID [0] 0.0.0.1 /service/ip 10433 0
```

PCM_OP_ORDER_POL_CREATE

Allows customized validation during order creation.

This opcode is called by the PCM_OP_ORDER_CREATE standard opcode during order creation and takes the same input as the PCM_OP_ORDER_CREATE standard opcode.

See the discussion on creating order objects in *BRM Developer's Guide*.

Example 1–228 Sample Input Flist

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order/voucher -1
0  PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0  PIN_FLD_ORDER_ID          STR [0] "12342"
0  PIN_FLD_STATE_ID          INT [0] 1
0  PIN_FLD_DESCR              STR [0] "Sample Order"
```

PCM_OP_ORDER_POL_DELETE

Allows customized validation during order deletion.

This opcode is called by the PCM_OP_ORDER_DELETE standard opcode and takes the same input as the PCM_OP_ORDER_DELETE standard opcode.

See the discussion on deleting order objects in *BRM Developer's Guide*.

Example 1–229 Sample Input Flist

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /order 11902 0
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
```

PCM_OP_ORDER_POL_PROCESS

Allows customized processing of the order response.

This opcode is called by the PCM_OP_ORDER_PROCESS standard opcode during order processing and takes the same input as the PCM_OP_ORDER_PROCESS standard opcode.

See the discussion on processing order response files in *BRM Developer's Guide*.

Example 1–230 Sample Input Flist

```
0  PIN_FLD_POID                POID [0] 0.0.0.1 /order 11249 0
0  PIN_FLD_PROGRAM_NAME        STR [0] "testnap"
0  PIN_FLD_ORDER_OLD_STATE     INT [0] 1
0  PIN_FLD_ORDER_NEW_STATE      INT [0] 2
```

Example 1–231 Sample Output Flist

```
0  PIN_FLD_POID                POID [0] 0.0.0.1 /order 11249 0
```

PCM_OP_ORDER_POL_SET_ATTR

Allows customized validation of order attribute changes.

This opcode is called by the PCM_OP_ORDER_SET_ATTR standard opcode during attribute changes and takes the same input as the PCM_OP_ORDER_SET_ATTR standard opcode.

See the discussion on changing order object attributes in *BRM Developer's Guide*.

Example 1–232 Sample Input Flist

```
0  PIN_FLD_POID             POID [0] 0.0.0.1 /order 8369 0
0  PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0  PIN_FLD_ORDER_ID         STR [0] "abcd"
0  PIN_FLD_DESCR             STR [0] "New order"
```

PCM_OP_ORDER_POL_SET_BRAND

Allows customized validation of brand changes of the order.

This opcode is called by the PCM_OP_ORDER_SET_BRAND standard opcode during brand changes and takes the same input as the PCM_OP_ORDER_SET_BRAND standard opcode.

See the discussion on changing order object brand associations in *BRM Developer's Guide*.

Example 1–233 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 17841 0
```

Example 1–234 Sample Output Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 10337 0
```

PCM_OP_ORDER_POL_SET_STATE

Allows customization during state changes.

This policy opcode can be called by PCM_OP_ORDER_SET_STATE standard opcode during state changes. Policy opcodes called during state changes are specified in the **/config/order_state** object.

Note: This opcode is supplied as the default policy for state changes.

See the discussion on setting the state in order objects in *BRM Developer's Guide*.

Example 1–235 Sample Input Flist

```
0  PIN_FLD_POID                POID [0] 0.0.0.1 /order 11249 0
0  PIN_FLD_ORDER_OLD_STATE    INT [0] 1
0  PIN_FLD_ORDER_NEW_STATE    INT [0] 2
```

Order FM Standard Opcodes

The opcodes described in Table 1–55 create, delete, and update **/order** objects.

Header File

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

Opcode Index

Table 1–55 Order FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ORDER_ASSOCIATE	Associates or disassociates an order with a master **/order** object. See the discussion on associating and disassociating order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_CREATE	Creates a new **/order** object. See the discussion on creating order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_DELETE	Deletes an **/order** object. See the discussion on deleting order objects in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_PROCESS	Processes the response of the order. See the discussion on processing order response files in *BRM Developer's Guide*.	Recommended
PCM_OP_ORDER_SET_ATTR	Sets attribute values for an **/order** object. See the discussion on changing order object attributes in *BRM Developer's Guide*.	Limited
PCM_OP_ORDER_SET_BRAND	Sets the brand for an **/order** object. See the discussion on changing order object brand associations in *BRM Developer's Guide*.	Limited
PCM_OP_ORDER_SET_STATE	Sets the state for an **/order** object. See the discussion on setting the state in order objects in *BRM Developer's Guide*.	Limited
PCM_OP_ORDER_UPDATE	Updates the state, brand, or attributes for an **/order** object. See the discussion on updating order objects in *BRM Developer's Guide*.	Recommended

PCM_OP_ORDER_ASSOCIATE

Associates or disassociates **/order** objects with a master object.

See the discussion on associating and disassociating order objects in *BRM Developer's Guide*.

Example 1–236 Sample Input Flist

Associating an order with a master order:

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0  PIN_FLD_FLAGS             INT [0] 1
0  PIN_FLD_ORDERS            ARRAY [0]
1  PIN_FLD_ORDER_OBJ          POID [0] 0.0.0.1 /order/num 10433 0
```

Disassociating an order from a master order:

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0  PIN_FLD_FLAGS             INT [0] 0
0  PIN_FLD_ORDERS            ARRAY [0]
1  PIN_FLD_ORDER_OBJ          POID [0] 0.0.0.1 /order/num 10433 0
```

Example 1–237 Sample Output Flist

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 10337 0
```

PCM_OP_ORDER_CREATE

Creates an **/order** object.

See the discussion on creating order objects in *BRM Developer's Guide*.

Example 1–238 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order/voucher -1
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_ORDER_ID        STR [0] "12342"
0  PIN_FLD_STATE_ID        INT [0] 1
0  PIN_FLD_DESCR            STR [0] "Sample Order"
```

Example 1–239 Sample Output Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 11249 0
```

PCM_OP_ORDER_DELETE

Deletes an **/order** object.

See the discussion on deleting order objects in *BRM Developer's Guide*.

Example 1–240 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 11902 0
0  PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
```

Example 1–241 Sample Output Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 11902 0
```

PCM_OP_ORDER_PROCESS

Processes the response to the order.

See the discussion on processing order response files in *BRM Developer's Guide*.

Example 1–242 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order/voucher 14549 1
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_ORDERS_INFO     ARRAY [0]
1  PIN_FLD_POID            POID [0] 0.0.0.1 /order/voucher 14549 1
1  PIN_FLD_SOURCE          STR [0] "SRC1"
1  PIN_FLD_MANUFACTURER    STR [0] "Clear Tech"
1  PIN_FLD_MODEL           STR [0] "PrePaid"
1  PIN_FLD_OBJ_TYPE        STR [0] "/voucher"
1  PIN_FLD_DEVICES         ARRAY [0]
2  PIN_FLD_POID            POID [0] 0.0.0.1 /device/voucher -1 0
2  PIN_FLD_DEVICE_ID        STR [0] "1101"
```

Example 1–243 Sample Output Flist

```
0  PIN_FLD_POID               POID [0] 0.0.0.1 /order 11249 0
```

PCM_OP_ORDER_SET_ATTR

Changes the attributes for an **/order** object.

See the discussion on changing order object attributes in *BRM Developer's Guide*.

Example 1–244 Sample Input Flist

```
0  PIN_FLD_POID             POID [0] 0.0.0.1 /order/voucher 8369 0
0  PIN_FLD_PROGRAM_NAME     STR [0] "testnap"
0  PIN_FLD_NAMEINFO         ARRAY [0]
1  PIN_FLD_ADDRESS           STR [0]
```

Example 1–245 Sample Output Flist

```
0  PIN_FLD_POID              POID [0] 0.0.0.1 /order 8369 0
```

PCM_OP_ORDER_SET_BRAND

Changes the brand association of the order.

See the discussion on changing order object brand associations in *BRM Developer's Guide*.

Example 1–246 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 10337 0
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 17841 0
```

Example 1–247 Sample Output Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order 10337 0
```

PCM_OP_ORDER_SET_STATE

Sets the state for an **/order** object.

See the discussion on setting the state in order objects in *BRM Developer's Guide*.

Example 1–248 Sample Input Flist

```
0  PIN_FLD_POID                 POID [0] 0.0.0.1 /order 11249 0
0  PIN_FLD_PROGRAM_NAME         STR [0] "testnap"
0  PIN_FLD_ORDER_OLD_STATE      INT [0] 1
0  PIN_FLD_ORDER_NEW_STATE       INT [0] 2
```

Example 1–249 Sample Output Flist

```
0  PIN_FLD_POID                 POID [0] 0.0.0.1 /order 11249 0
```

PCM_OP_ORDER_UPDATE

Updates the attributes, state, or brand for an **/order** object.

See the discussion on updating order objects in *BRM Developer's Guide*.

Example 1–250 Sample Input Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order/voucher 14549 0
0  PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0  PIN_FLD_STATE_ID         INT [0] 3
```

Example 1–251 Sample Output Flist

```
0  PIN_FLD_POID            POID [0] 0.0.0.1 /order/voucher 14549 0
```

Offer Profile FM Standard Opcodes

The opcodes listed in Table 1–56 create and manage the offer profile. See the discussion on managing offer profiles with offer profile opcodes in *BRM Setting Up Pricing and Rating*.

Header File

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

Opcode Index

Table 1–56 Offer Profile FM Standard Opcodes

Opcode	Description	Use
PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD	Based on the offer profile passed in the input flist, gets the dynamic information for a service POID and the resource ID. See the discussion on readjusting quota for requests to update and authorize in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE	Creates, modifies, or deletes the offer profiles (**/offer_profile** objects). See the discussion on the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode in *BRM Setting Up Pricing and Rating*.	Limited

PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD

For policy-driven charging, this opcode uses the offer profiles passed in the input flist to retrieve the dynamic information for a given service POID and resource ID.

When the balances (without current impact) are passed in the BALANCES array of the opcode's input flist, the opcode returns if any threshold breach notifications result from the balance impacts due to the resource usage.

When the additional quota reported by the network is passed in the QUANTITY field of the opcode's input flist, the opcode readjusts the quota for requests to update and authorize.

See the following discussions in *BRM Setting Up Pricing and Rating*:

- Readjusting quota for requests to update and authorize
- Determining if balance impacts trigger notifications

PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE

Creates, modifies, and deletes offer profiles.

See the discussion on the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode in *BRM Setting Up Pricing and Rating*.

Permissioning FM Standard Opcodes

The opcodes listed in Table 1–57 create and manage Access Control Lists (ACLs), which specify the CSRs that can access customer accounts in a brand or account group. See the discussion on configuring a branded database in *BRM Managing Customers*.

Header File

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

Opcode Index

Table 1–57 Permissioning FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PERM_ACL_GET_SUBGROUPS	Retrieves a list of billing subgroups.	Limited
PCM_OP_PERM_ACL_GROUP_CREATE	Creates a **/group/acl** object.	Limited
PCM_OP_PERM_ACL_GROUP_DELETE	Deletes a **/group/acl** object.	Limited
PCM_OP_PERM_ACL_GROUP_MODIFY	Modifies a **/group/acl** object.	Limited
PCM_OP_PERM_ACL_GROUP_ADD_MEMBER	Adds a group member to a **/group/acl**.	Limited
PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER	Deletes a group member from a **/group/acl**	Limited
PCM_OP_PERM_FIND	Finds user authorized **/group/acl**.	Limited
PCM_OP_PERM_GET_CREDENTIALS	Retrieves a list of brands.	Limited
PCM_OP_PERM_SET_CREDENTIALS	Sets the connection scope to a brand.	Limited

PCM_OP_PERM_ACL_GET_SUBGROUPS

Retrieves a particular **/group/billing** hierarchy beneath the provided access control list.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_ACL_GROUP_CREATE

Creates a **/group/acl** object.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_ACL_GROUP_DELETE

Deletes a **/group/acl** object.

Important: Deleting an ACL does not delete the brand account or affect services. It simply removes the ACL. This opcode is used when an administrator removes existing service authorizations for a brand account.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_ACL_GROUP_MODIFY

Modifies the attributes in a **/group/acl** object.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_ACL_GROUP_ADD_MEMBER

Adds group members to a **/group/acl** object.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER

Deletes a member from a **/group/acl** object.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_FIND

Retrieves a list of Access Control Lists (ACLs) to which a CSR belongs and returns user specified information about each ACL.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_GET_CREDENTIALS

Retrieves a list of ACLs, brand accounts, and billing groups that can access an application. This opcode is useful to developers of multi-brand applications who often need to know which brands have access to an application.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

PCM_OP_PERM_SET_CREDENTIALS

Sets the current connection scope to a brand. Scope is defined by two parameters: a brand to be activated and optionally, the billing group.

See the discussion on managing permission by using a custom application in *BRM Managing Customers*.

Price List FM Policy Opcodes

The opcodes listed in Table 1–58 create, delete, and modify price list components, including plans, deals, products, and rates.

Header File

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

Opcode Index

Table 1–58 Price List FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PRICE_POL_DELETE_DEAL	Verifies that deleting a **/deal** object is permitted. See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_DELETE_DEPENDENCY	Verifies that deleting a **/dependency** object is permitted. See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_DELETE_DISCOUNT	Verifies that deleting a **/discount** object is permitted. See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_DELETE_PRODUCT	Verifies that deleting a **/product** object is permitted. See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_DELETE_TRANSITION	Verifies that deleting a **/transition** object is permitted. See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_PREP_DEAL	Passes the values for a **/deal** object. See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_PREP_DEPENDENCY	Allows data modification during **/dependency** object creation. See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.	Recommended

Table 1–58 (Cont.) Price List FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PRICE_POL_PREP_DISCOUNT	Allows data modification during **/discount** object creation. See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_PREP_PRODUCT	Passes the values for a **/product** object. See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_PREP_TRANSITION	Allows data modification during **/transition** object creation. See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_VALID_DEAL	Allows validation during **/deal** object creation. See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_VALID_DEPENDENCY	Allows validation during **/dependency** object creation. See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_VALID_DISCOUNT	Allows validation during **/discount** object creation. See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_VALID_PRODUCT	Passes the input fields for a new or changed **/product** object. See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_POL_VALID_TRANSITION	Allows validation during **/transition** object creation. See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_PRICE_POL_DELETE_DEAL

Verifies that deleting a **/deal** object is permitted.

Use this opcode to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DEAL standard opcode.

See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_DELETE_DEPENDENCY

Verifies that deleting a **/dependency** object is permitted.

Use this opcode to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DEPENDENCY standard opcode.

See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_DELETE_DISCOUNT

Verifies that deleting a **/discount** object is permitted.

Performs validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DISCOUNT standard opcode.

See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_DELETE_PRODUCT

Verify that deleting a **/product** object is permitted.

Use this opcode to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_PRODUCT standard opcode.

See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_DELETE_TRANSITION

Verifies that deleting a **/transition** object is permitted.

Use this opcode to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_TRANSITION standard opcode.

See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_PREP_DEAL

Use this opcode to enhance **/deal** objects with additional data not provided by either the GUI application or by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DEAL standard opcode.

See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_PREP_DEPENDENCY

Use this opcode to enhance **/dependency** objects with additional data not provided by either the GUI application or by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DEPENDENCY standard opcode.

See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_PREP_DISCOUNT

Use this opcode to enhance **/discount** objects with additional data not provided by either the GUI application or by the Price List FM.

This opcode is called by the PCM_OPPRICE_COMMIT_SPONSORSHIP standard opcode.

See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_PREP_PRODUCT

Use this opcode to enhance **/product** objects with additional data not provided by either the GUI application or by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_PRODUCT standard opcode.

See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_PREP_TRANSITION

Use this opcode to enhance **/transition** objects with additional data not provided by either the GUI application or by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_TRANSITION standard opcode.

See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_VALID_DEAL

Validates data during **/deal** object creation.

This policy opcode can be used to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_DEAL standard opcode.

See the discussion on customizing how to create and delete deals in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_VALID_DEPENDENCY

Validates data during **/dependency** object creation.

This policy opcode can be used to change **/dependency** relationships without using Pricing Center.

This opcode is called by the PCM_OP_PRICE_SET_PRICE_LIST and PCM_OP_PRICE_COMMIT_DEPENDENCY standard opcodes.

See the discussion on customizing how to create and delete dependencies in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_VALID_DISCOUNT

Validates data during **/discount** object creation.

This policy opcode can be used to enhance **/discount** objects with additional data not provided by either Pricing Center or by other opcodes in the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_TRANSITION standard opcode.

See the discussion on customizing how to create and delete discounts in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_VALID_PRODUCT

Validates data during **/product** object creation.

This policy opcode can be used to perform validations in addition to those performed by the Price List FM.

This opcode is called by the PCM_OP_PRICE_COMMIT_PRODUCT standard opcode.

See the discussion on customizing how to create and delete products in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_POL_VALID_TRANSITION

Validates data during **/transition** object creation.

This policy opcode can be used to change **/transition** relationships without using Pricing Center.

This policy opcode is called by the PCM_OP_PRICE_COMMIT_TRANSITION standard opcode.

See the discussion on customizing how to create and delete transitions in *BRM Setting Up Pricing and Rating*.

Price List FM Standard Opcodes

The opcodes listed in Table 1–59 create, delete, and modify price list components, such as plans, deals, products, and rates.

Header File

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

Opcode Index

Table 1–59 Price List FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PRICE_COMMIT_DEAL	Creates, changes, or deletes a **/deal** object. See the discussion on managing deal objects in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_PRICE_COMMIT_DEPENDENCY	Creates, changes, or deletes a **/dependency** object. See the discussion on managing dependency objects in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_COMMIT_DISCOUNT	Creates, changes, or deletes a **/discount** object. See the discussion on managing discount objects in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_PRICE_COMMIT_PLAN	Creates, changes, or deletes a **/plan** object. See the discussion on managing plan objects in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_PRICE_COMMIT_PLAN_LIST	Validates and commits a **/group/plan_list** object into the database. See the discussion on managing group plan_list objects in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_PRICE_COMMIT_PRODUCT	Validates and commits **/product** objects. See the discussion on managing product objects in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_PRICE_COMMIT_SPONSORSHIP	Creates, changes, or deletes a **/sponsorship** object. See the discussion on managing sponsorship objects in *BRM Setting Up Pricing and Rating*.	Limited

Table 1–59 (Cont.) Price List FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PRICE_COMMIT_TRANSITION	Creates, changes, or deletes a **/transition** object. See the discussion on managing transition objects in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_GET_DISCOUNT_INFO	Retrieves real-time discount data along with pipeline discount model data. See the discussion on retrieving discount data in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_GET_PRICE_LIST	Retrieves pricing objects from the BRM database. See the discussion on retrieving price list data from the BRM database in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_PRICE_GET_PRODUCT_INFO	Retrieves information about the product specified in the input flist, including pipeline rate plan information and provisioning tag information. See the discussion on retrieving product details in *BRM Managing Customers*.	Recommended
PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT	Assembles an flist for creating a customized **/product** object. See the discussion on creating customized product objects in *BRM Managing Customers*.	Recommended
PCM_OP_PRICE_SET_PRICE_LIST	Commits pricing objects to the database. See the discussion on committing price list data to the BRM database in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_PRICE_COMMIT_DEAL

Creates, changes, or deletes a **/deal** object.

Important: This opcode overwrites data in existing **/deal** objects, so be sure you pass in the correct object to modify.

See the discussion on managing deal objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_DEPENDENCY

Creates, modifies, or deletes a **/dependency** object.

See the discussion on managing dependency objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_DISCOUNT

Creates, changes, or deletes a **/discount** object.

Important: This opcode overwrites data in existing **/discount** objects, so be sure you pass in the correct object to modify.

See the discussion on managing discount objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_PLAN

Creates, changes, or deletes a **/plan** object.

Important: This opcode overwrites data in existing **/plan** objects, so be sure you pass in the correct object to modify.

See the discussion on managing plan objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_PLAN_LIST

Commits the **/group/plan_list** object to the database.

See the discussion on managing group plan_list objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_PRODUCT

Commits product pricing information to the database.

This opcode validates and commits the following objects from a product flist: **/rate**, **/rate_plan**, **/rate_plan_selector**, and **/rollover**. Products are created or modified and, if the delete flag is sent in, deleted.

This opcode also publishes an **/event/notification/price/products/modify** notification event.

See the discussion on managing product objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_SPONSORSHIP

Creates, changes, or deletes a **/sponsorship** object.

Important: This opcode overwrites data in existing **/sponsorship** objects, so be sure you pass in the correct object to modify.

See the discussion on managing sponsorship objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_COMMIT_TRANSITION

Creates, changes, or deletes a **/transition** object.

See the discussion on managing transition objects in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_GET_DISCOUNT_INFO

Retrieves real-time discount data along with pipeline discount model data from the BRM database. The discount model information retrieved includes the following:

- Discount model version and configuration
- Discount/ chargeshare trigger
- Discount/ chargeshare condition
- Discount/ chargeshare rules
- Discount/ chargeshare master
- Discount/ chargeshare detail
- Discount/ chargeshare step
- Balance impact

See the discussion on retrieving discount data in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_GET_PRICE_LIST

Retrieves pricing objects from the BRM database.

See the discussion on retrieving price list data from the BRM database in *BRM Setting Up Pricing and Rating*.

PCM_OP_PRICE_GET_PRODUCT_INFO

Retrieves information about the product specified in the input flist, including:

- Pipeline rate plan information, if the product includes events configured for pipeline rating.
- Provisioning tag details from the **/config/provisioning_tag** object (for nontelco services) or the **/config/telco/*** object (for telco services), if a product is configured with a provisioning tag.

The opcode retrieves information from the **/product** object specified in the input flist. It also retrieves the content of **/rate_plan_selector**, **/rate_plan**, **/rate**, and **/rollover** objects contained in the **/product** object.

This opcode analyzes the usage maps in the **/product** object. If a usage map array includes an event to be rated by a pipeline rate plan, the opcode optionally retrieves pipeline rate plan data from the database. You specify whether the opcode retrieves pipeline rate plan data by passing the FM_PRICE_SUPPRESS_PIPELINE_DATA flag in the opcode call.

See the following discussions:

- Retrieving product details in *BRM Managing Customers* for more information about FM_PRICE_SUPPRESS_PIPELINE_DATA flag
- Understanding the PCM API and the PIN library in *BRM Developer's Guide* for more information about passing flags in opcode calls
- Working with provisioning tags in *BRM Setting Up Pricing and Rating*

PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT

This opcode assembles an flist for creating or modifying a customized **/product** object. The opcode calculates new rates for the product based on the list of resources and override percentages in the PIN_FLD_TAILORMADE_DATA field in the input flist.

The input flists for creating or modifying a customized **/product** object are similar except for two fields:

- The PIN_FLD_POID field in the PIN_FLD_PRODUCTS array is a type-only POID when creating a customized product. When modifying a customized product, it is the POID of the existing customized **/product** object.
- The PIN_FLD_NAME field in the PIN_FLD_PRODUCTS array is the name of the base product when creating a customized product. When modifying a customized product, it is the name of the existing customized **/product** object.

For new customized **/product** objects, the opcode assigns a name to the customized product object by prepending the current time in seconds, represented in hexadecimal, to the base product name. For example, if the base product name is **StandardGSMTelephony**, the generated name for the customized product might be **AE9C6890_StandardGSMTelephony**.

After processing, the opcode returns a complete customized **/product** flist. You pass this flist to PCM_OP_PRICE_SET_PRICE_LIST.

The opcode is called by Customer Center or another CRM applications during the creation or modification of a customized product.

Example 1–252 Sample Input Flists

The following input produces an flist for a **/product** object with a customized real-time rate plan.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /product 58676 0
0 PIN_FLD_PRODUCTS      ARRAY [0] allocated 25, used 25
1     PIN_FLD_POID           POID [0] 0.0.0.1 /product -1 1
1     PIN_FLD_CREATED_T    TSTAMP [0] (1157359113) Mon Sep  4 14:08:33 2006
1     PIN_FLD_MOD_T        TSTAMP [0] (1157359113) Mon Sep  4 14:08:33 2006
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_BASE_PRODUCT_OBJ   POID [0] 0.0.0.1 /product 58676 1
1     PIN_FLD_DESCR           STR [0] "Testing"
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_NAME            STR [0] "TMP_Prod_111"
1     PIN_FLD_OWN_MAX      DECIMAL [0] NULL
1     PIN_FLD_OWN_MIN      DECIMAL [0] NULL
1     PIN_FLD_PARTIAL        ENUM [0] 0
1     PIN_FLD_PERMITTED       STR [0] "/service/ip"
1     PIN_FLD_PRIORITY     DECIMAL [0] 0
1     PIN_FLD_PROVISIONING_TAG    STR [0] ""
1     PIN_FLD_PURCHASE_MAX DECIMAL [0] NULL
1     PIN_FLD_PURCHASE_MIN DECIMAL [0] NULL
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_TAILORMADE      INT [0] 0
1     PIN_FLD_TAX_SUPPLIER    INT [0] 0
1     PIN_FLD_TYPE           ENUM [0] 602
1     PIN_FLD_ZONEMAP_NAME    STR [0] ""
1     PIN_FLD_USAGE_MAP     ARRAY [0] allocated 20, used 12
2         PIN_FLD_EVENT_TYPE      STR [0] "/event/session"
```

```
2         PIN_FLD_FLAGS           INT [0] 0
2         PIN_FLD_INCR_QUANTITY DECIMAL [0] 1
2         PIN_FLD_INCR_UNIT      ENUM [0] 0
2         PIN_FLD_MIN_QUANTITY DECIMAL [0] 1
2         PIN_FLD_MIN_UNIT       ENUM [0] 0
2         PIN_FLD_RATE_PLAN_NAME    STR [0] "Session Event"
2         PIN_FLD_RATE_PLAN_SELECTOR_OBJ   POID [0] 0.0.0.0  0 0
2         PIN_FLD_ROUNDING_MODE   ENUM [0] 0
2         PIN_FLD_RUM_NAME        STR [0] "Duration"
2         PIN_FLD_TIMEZONE_MODE   ENUM [0] 0
2         PIN_FLD_TOD_MODE       ENUM [0] 0
1     PIN_FLD_RATE_PLANS    ARRAY [0] allocated 20, used 17
2         PIN_FLD_POID           POID [0] 0.0.0.1 /rate_plan 58164 0
2         PIN_FLD_CREATED_T    TSTAMP [0] (1157359113) Mon Sep  4 14:08:33 2006
2         PIN_FLD_MOD_T        TSTAMP [0] (1157359113) Mon Sep  4 14:08:33 2006
2         PIN_FLD_READ_ACCESS     STR [0] "B"
2         PIN_FLD_WRITE_ACCESS    STR [0] "S"
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
2         PIN_FLD_BILL_OFFSET     INT [0] 0
2         PIN_FLD_CURRENCY        INT [0] 840
2         PIN_FLD_CYCLE_FEE_FLAGS    INT [0] 0
2         PIN_FLD_EVENT_TYPE      STR [0] "/event/session"
2         PIN_FLD_NAME            STR [0] "Session Event"
2         PIN_FLD_OFFSET_UNIT    ENUM [0] 0
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 58676 0
2         PIN_FLD_TAX_CODE        STR [0] ""
2         PIN_FLD_TAX_WHEN       ENUM [0] 0
2         PIN_FLD_RATE_TIERS    ARRAY [0] allocated 20, used 5
3             PIN_FLD_DATE_RANGE_TYPE   ENUM [0] 0
3             PIN_FLD_NAME            STR [0] "Tier 1"
3             PIN_FLD_PRIORITY     DECIMAL [0] 0
3             PIN_FLD_RATE_OBJ       POID [0] 0.0.0.1 /rate 11111 0
3             PIN_FLD_RATE_INDEX      INT [0] 0
2         PIN_FLD_RATES         ARRAY [0] allocated 20, used 15
3             PIN_FLD_POID           POID [0] 0.0.0.1 /rate 11111 1
3             PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
3             PIN_FLD_DESCR           STR [0] "Rate 1"
3             PIN_FLD_PRORATE_FIRST   ENUM [0] 702
3             PIN_FLD_PRORATE_LAST   ENUM [0] 701
3             PIN_FLD_RATE_PLAN_OBJ   POID [0] 0.0.0.1 /rate_plan 58164 0
3             PIN_FLD_STEP_RESOURCE_ID    INT [0] 0
3             PIN_FLD_STEP_TYPE      ENUM [0] 0
3             PIN_FLD_TAILORMADE_DATA    STR [0] "USD,-20"
```

The following input produces an flist for a **/product** object with a customized pipeline rate plan.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /product 59229 0
0 PIN_FLD_PRODUCTS      ARRAY [0] allocated 20, used 4
1     PIN_FLD_POID           POID [0] 0.0.0.1 /product -1 1
1     PIN_FLD_NAME            STR [0] "TMP_pipe_111"
1     PIN_FLD_BASE_PRODUCT_OBJ   POID [0] 0.0.0.1 /product $(product_poid) 0
1     PIN_FLD_PIPELINE_RATEPLANS  ARRAY [1] allocated 20, used 14
2         PIN_FLD_RATE_PLAN_NAME    STR [0] "Standard"
2         PIN_FLD_SEQUENCE_NUM    INT [0] 20001
2         PIN_FLD_RATE_PLAN_CODE    STR [0] "Standard"
2         PIN_FLD_STATUS_STR      STR [0] "A"
2         PIN_FLD_MODEL_TYPE      STR [0] "R"
2         PIN_FLD_SPLITTING       STR [0] "1"
2         PIN_FLD_CALENDAR        INT [0] 20000
2         PIN_FLD_UTC_TIMEOFFSET    STR [0] "+0100"
```

```
2         PIN_FLD_CURRENCY_NAME    STR [0] "EUR"
2         PIN_FLD_TAX_TREATMENT    INT [0] 0
2         PIN_FLD_TAILORMADE      INT [0] 0
2         PIN_FLD_CALENDAR_CODE    STR [0] "ALL_RATE"
2         PIN_FLD_CALENDAR_NAME    STR [0] "Wireless Sample Calendar"
2         PIN_FLD_RATEPLAN_VER  ARRAY [1] allocated 20, used 11
3             PIN_FLD_VERSION_ID      INT [0] 1
3             PIN_FLD_VALID_FROM   TSTAMP [0] (915148800) Fri Jan  1 05:30:00 1999
3             PIN_FLD_STATUS_STR      STR [0] "A"
3             PIN_FLD_ZONEMODEL       INT [0] 20000
3             PIN_FLD_BASIC           INT [0] 1
3             PIN_FLD_ZONEMODEL_CODE    STR [0] "ALL_RATE"
3             PIN_FLD_ZONEMODEL_NAME    STR [0] "Wireless Sample ZoneModel to be
used for all services (TEL, SMS, GPRS, WAP)"
3             PIN_FLD_RATEPLAN_CNF  ARRAY [3] allocated 20, used 16
4                 PIN_FLD_VERSION_ID      INT [0] 1
4                 PIN_FLD_SVC_CODE        STR [0] "TEL"
4                 PIN_FLD_SVC_CLASS       STR [0] "DEF"
4                 PIN_FLD_IMPACT_CATEGORY    STR [0] "EUROPE"
4                 PIN_FLD_TIMEMODEL_INT    INT [0] 20003
4                 PIN_FLD_TIMEZONE        INT [0] 20003
4                 PIN_FLD_ADDON_TYPE      STR [0] "P"
4                 PIN_FLD_ADDON_CHARGE DECIMAL [0] 0
4                 PIN_FLD_PASSTHROUGH     INT [0] 0
4                 PIN_FLD_PRICE_MODEL_INDEX    INT [0] 16
4                 PIN_FLD_SVC_CODE_NAME    STR [0] "Telephony"
4                 PIN_FLD_IMP_CAT_NAME    STR [0] "Usage within Europe (outside
0049)"
4                 PIN_FLD_TIMEMODEL_CODE    STR [0] "EUROPEAN"
4                 PIN_FLD_TIMEMODEL_NAME    STR [0] "European TimeModel"
4                 PIN_FLD_TIMEZONE_CODE    STR [0] "WEEKOFF2"
4                 PIN_FLD_TIMEZONE_NAME    STR [0] "Weekdays OffPeak 2"
3             PIN_FLD_RATEPLAN_CNF  ARRAY [45] allocated 20, used 16
4                 PIN_FLD_VERSION_ID      INT [0] 1
4                 PIN_FLD_SVC_CODE        STR [0] "TEL"
4                 PIN_FLD_SVC_CLASS       STR [0] "DEF"
4                 PIN_FLD_IMPACT_CATEGORY    STR [0] "NAT_PREM"
4                 PIN_FLD_TIMEMODEL_INT    INT [0] 20001
4                 PIN_FLD_TIMEZONE        INT [0] 20003
4                 PIN_FLD_ADDON_TYPE      STR [0] "P"
4                 PIN_FLD_ADDON_CHARGE DECIMAL [0] 0
4                 PIN_FLD_PASSTHROUGH     INT [0] 0
4                 PIN_FLD_PRICE_MODEL_INDEX    INT [0] 14
4                 PIN_FLD_SVC_CODE_NAME    STR [0] "Telephony"
4                 PIN_FLD_IMP_CAT_NAME    STR [0] "National Premium Calls"
4                 PIN_FLD_TIMEMODEL_CODE    STR [0] "NATIONAL"
4                 PIN_FLD_TIMEMODEL_NAME    STR [0] "National TimeModel"
4                 PIN_FLD_TIMEZONE_CODE    STR [0] "WEEKOFF2"
4                 PIN_FLD_TIMEZONE_NAME    STR [0] "Weekdays OffPeak 2"
3             PIN_FLD_PRICE_MODELS  ARRAY [14] allocated 20, used 4
4                 PIN_FLD_PRICE_MODEL_CODE    STR [0] "T1.99_60"
4                 PIN_FLD_SEQUENCE_NUM    INT [0] 20030
4                 PIN_FLD_PRICE_MODEL_NAME    STR [0] "TEL 1.99 EUR, beat: 60"
4                 PIN_FLD_TAILORMADE_DATA    STR [0] "EURO,-10"
3             PIN_FLD_PRICE_MODELS  ARRAY [16] allocated 20, used 4
4                 PIN_FLD_PRICE_MODEL_CODE    STR [0] "T0.10_60"
4                 PIN_FLD_SEQUENCE_NUM    INT [0] 20036
4                 PIN_FLD_PRICE_MODEL_NAME    STR [0] "TEL 0.10 EUR, beat: 60"
4                 PIN_FLD_TAILORMADE_DATA    STR [0] "EURO,-20"
```

PCM_OP_PRICE_SET_PRICE_LIST

Creates, modifies, or deletes the following price list data in the BRM database in a single transaction: **/best_pricing**, **/discount**, **/group/plan_list**, **/plan**, **/deal**, **/product**, **/dependency**, **/transition**, and **/sponsorship**.

See the discussion on committing price list data to the BRM database in *BRM Setting Up Pricing and Rating*.

Process Audit FM Policy Opcodes

The opcodes in Table 1–60 are called by billing utilities and Pipeline Manager 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 Configuring and Running Billing.

Header File

Includes the *BRM_Home*/**include/ops/process_audit.h** header file in all applications that call these opcodes. See the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–60 Process Audit FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PROCESS_AUDIT_POL_CREATE	Gets fields from **/config objects**, checks for duplicate fields, and validates the data. See the discussion on customizing audit object validation in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_POL_CREATE_AND_LINK	Checks for duplicate objects of **/process_audit/batchstat** storable class. See the discussion on customizing **/process_audit/batchstat** object validation in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_POL_ALERT	Sends email messages when configuration thresholds are crossed. See the discussion on customizing alert behavior in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY	Maps the fields of **/suspended_usage/telco** to the corresponding fields of **/process_audit/batchstat** storable class. See the discussion on customizing the revenue assurance written-off event data record (EDR) summaries in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_PROCESS_AUDIT_POL_CREATE

Checks for duplication of 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 the PCM_OP_PROCESS_AUDIT_CREATE standard opcode.

See the discussion on customizing audit object validation in BRM Configuring and Running Billing.

Transaction Handling

The transaction handling for this opcode is done in the standard opcode.

PCM_OP_PROCESS_AUDIT_POL_CREATE_AND_LINK

Checks for duplication of audit objects and entries.

You can customize this opcode by modifying the fields in the flist, modifying duplicate checks, and adding validation checks.

This opcode is called by the PCM_OP_PROCESS_AUDIT_CREATE_AND_LINK standard opcode.

See the discussion on customizing process_audit batchstat object validation in BRM Configuring and Running Billing.

Transaction Handling

The transaction handling for this opcode is done in the standard opcode.

PCM_OP_PROCESS_AUDIT_POL_ALERT

Sends email messages when configured 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.

See the discussion on customizing alert behavior in BRM Configuring and Running Billing.

PCM_OP_PROCESS_AUDIT_POL_CREATE_WRITEOFF_SUMMARY

Called by PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY opcode to map the fields of **/suspended_usage/telco** storable class to the corresponding fields of **/process_audit/batchstat/status** storable class.

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 **/process_audit/batchstat/status** storable class.

This opcode is called by the PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY standard opcode.

See the discussion on customizing the revenue assurance written-off EDR summaries in BRM Configuring and Running Billing.

Transaction Handling

The transaction handling for this opcode is done in the standard opcode.

Process Audit FM Standard Opcodes

The opcodes listed in Table 1–61 are called by billing utilities and Pipeline Manager 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 Configuring and Running Billing.

Header File

Includes the *BRM_Home*/**include/ops/process_audit.h** header file in all applications that call these opcodes. See the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–61 Process Audit FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PROCESS_AUDIT_CREATE	*Creates* **/process_audit/pipeline** and **/process_audit billing** audit objects. See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_CREATE_AND_LINK	Creates **/process_audit/batchstat** objects. See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_LINK	Links **/process_audit/batchstat** objects. See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY	Creates **/process_audit/batchstat/status** objects with revenue assurance data for written-off EDRs. 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. See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_PROCESS_AUDIT_SEARCH	Retrieves summary and detail data for control points.	Recommended

PCM_OP_PROCESS_AUDIT_CREATE

Creates audit objects for revenue assurance.

Called by the BRM billing applications and the Universal Event Loader.

See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.

PCM_OP_PROCESS_AUDIT_CREATE_AND_LINK

Creates **/process_audit/batchstat** objects.

See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.

PCM_OP_PROCESS_AUDIT_LINK

Links the **/process_audit/batchstat** objects according to the specified configuration.

See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.

PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY

Creates a summary of the written-off EDRs.

See the discussion on the Revenue Assurance Manager opcodes in *BRM Configuring and Running Billing*.

PCM_OP_PROCESS_AUDIT_SEARCH

Retrieves summary and detail data for control points for **/process_audit/batchstat/batchstat** objects and its subclasses.

This opcode returns an error message if the control point name or batch type is invalid.

See the discussion on the Revenue Assurance Manager opcodes in the BRM documentation in *BRM Configuring and Running Billing*.

Provisioning FM Policy Opcode

Use the opcode in Table 1–62 to customize provisioning.

Header File

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

Opcode Index

Table 1–62 Provisioning FM Policy Opcode

Opcode	Description	Use
PCM_OP_PROV_POL_UPDATE_SVC_ORDER	Policy for updating service orders. See the discussion on managing GSM service provisioning in *BRM Managing Customers*.	Recommended

PCM_OP_PROV_POL_UPDATE_SVC_ORDER

Validates and modifies parameters for updating service orders.

The input flist to this opcode includes the complete response from the provisioning applications. Based on the type of the service order, you can modify or validate the response flist.

This opcode is called by the PCM_OP_PROV_UPDATE_SVC_ORDER standard opcode when a response is received from a provisioning system.

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

Example 1–253 Sample Input Flist

This sample shows an input flist for a GSM service:

```
0 PIN_FLD_POID           POID 0x100 [0] 0.0.0.1 /event/provisioning/service_
order/telco/gsm 175921 0
0 PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 10
1   PIN_FLD_PARAMS        ARRAY [0] allocated 20, used 5
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
2     PIN_FLD_NAME            STR 0x100 [0] "VM"
2     PIN_FLD_STATUS         ENUM [0] 0
2     PIN_FLD_VALUE           STR 0x100 [0] ""
1   PIN_FLD_PARAMS        ARRAY [1] allocated 20, used 5
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
2     PIN_FLD_NAME            STR 0x100 [0] "CLID"
2     PIN_FLD_STATUS         ENUM [0] 0
2     PIN_FLD_VALUE           STR 0x100 [0] ""
1   PIN_FLD_PARAMS        ARRAY [7] allocated 20, used 5
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
2     PIN_FLD_NAME            STR 0x100 [0] "SIM"
2     PIN_FLD_STATUS         ENUM [0] 0
2     PIN_FLD_VALUE           STR 0x100 [0] "240192"
1 PIN_FLD_PARAMS        ARRAY [8] allocated 20, used 5
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
2     PIN_FLD_NAME            STR 0x100 [0] "IMSI"
2     PIN_FLD_STATUS         ENUM [0] 0
2     PIN_FLD_VALUE           STR 0x100 [0] ""
1 PIN_FLD_PARAMS        ARRAY [9] allocated 20, used 5
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
2     PIN_FLD_NAME            STR 0x100 [0] "MSISDN"
2     PIN_FLD_STATUS         ENUM [0] 0
2     PIN_FLD_VALUE           STR 0x100 [0] "0014085722000".
```

Example 1–254 Sample Output Flist

This sample shows an output flist for a GSM service:

```
0 PIN_FLD_POID           POID 0x100 [0] 0.0.0.1 /event/provisioning/service_
order/telco/gsm 175921 0
0 PIN_FLD_SERVICE_ORDER_INFO  ARRAY [0] allocated 10, used 10
1   PIN_FLD_POID           POID 0x100 [0] 0.0.0.1 /service/telco/gsm/telephony
12524 4
1   PIN_FLD_ACTION          STR 0x100 [0] "A"
1   PIN_FLD_NAME            STR 0x100 [0] ""
1   PIN_FLD_PARAMS        ARRAY [0] allocated 2, used 2
2     PIN_FLD_NAME            STR 0x100 [0] "VM"
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
```

```
1   PIN_FLD_PARAMS        ARRAY [1] allocated 2, used 2
2     PIN_FLD_NAME            STR 0x100 [0] "CLID"
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
0 PIN_FLD_SERVICE_ORDER_INFO  ARRAY [1] allocated 5, used 5
1   PIN_FLD_POID           POID 0x100 [0] 0.0.0.1 /device/sim 8763 1
1   PIN_FLD_ACTION          STR 0x100 [0] "A"
1   PIN_FLD_NAME            STR 0x100 [0] ""
1   PIN_FLD_PARAMS        ARRAY [0] allocated 2, used 2
2     PIN_FLD_NAME            STR 0x100 [0] "SIM"
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
1   PIN_FLD_PARAMS        ARRAY [1] allocated 2, used 2
2     PIN_FLD_NAME            STR 0x100 [0] "IMSI"
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
0 PIN_FLD_SERVICE_ORDER_INFO  ARRAY [2] allocated 4, used 4
1   PIN_FLD_POID           POID 0x100 [0] 0.0.0.1 /device/num 8529 1
1   PIN_FLD_ACTION          STR 0x100 [0] "A"
1   PIN_FLD_NAME            STR 0x100 [0] ""
1   PIN_FLD_PARAMS        ARRAY [0] allocated 2, used 2
2     PIN_FLD_NAME            STR 0x100 [0] "MSISDN"
2     PIN_FLD_ACTION          STR 0x100 [0] "A"
```

Example 1–255 Sample Input Flist

This sample shows an input flist for a GSM device service order with format transformed for BRM:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /event/provisioning/service_
order/telco/gsm 17592186111335 0
0 PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 5
1     PIN_FLD_PARAMS        ARRAY [0] allocated 20, used 4
2         PIN_FLD_NAME            STR [0] "SIM"
2         PIN_FLD_ACTION          STR [0] "I"
2         PIN_FLD_STATUS         ENUM [0] 0
1     PIN_FLD_PARAMS        ARRAY [1] allocated 20, used 4
2         PIN_FLD_NAME            STR [0] "MSISDN"
2         PIN_FLD_ACTION          STR [0] "I"
2         PIN_FLD_STATUS         ENUM [0] 0
1     PIN_FLD_PARAMS        ARRAY [2] allocated 20, used 4
2         PIN_FLD_NAME            STR [0] "IMSI"
2         PIN_FLD_ACTION          STR [0] "I"
2         PIN_FLD_STATUS         ENUM [0] 0
1     PIN_FLD_PARAMS        ARRAY [3] allocated 20, used 4
2         PIN_FLD_NAME            STR [0] "KI"
2         PIN_FLD_ACTION          STR [0] "I"
2         PIN_FLD_STATUS         ENUM [0] 0
1     PIN_FLD_PARAMS        ARRAY [4] allocated 20, used 4
2         PIN_FLD_NAME            STR [0] "NET"
2         PIN_FLD_ACTION          STR [0] "I"
2         PIN_FLD_STATUS         ENUM [0] 0
```

Example 1–256 Sample output list

This shows a sample output flist for a GSM device service order:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /event/provisioning/service_
order/telco/gsm 17592186111335 0
0 PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_SERVICE_ORDER_INFO  ARRAY [0] allocated 20, used 3
2         PIN_FLD_POID           POID [0] 0.0.0.1 /device/sim 67175 0
2         PIN_FLD_ACTION          STR [0] "P"
2         PIN_FLD_NAME            STR [0]
```

Provisioning FM Standard Opcodes

The opcodes in Table 1–63 manage service order provisioning.

Header File

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

Opcode Index

Table 1–63 Provisioning FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PROV_PUBLISH_SVC_ORDER	Publishes a service order. See the discussion on managing GSM service provisioning in *BRM Managing Customers*.	Recommended
PCM_OP_PROV_UPDATE_SVC_ORDER	Updates a service order. See the discussion on managing GSM service provisioning in *BRM Managing Customers*.	Recommended

PCM_OP_PROV_PUBLISH_SVC_ORDER

Sends a **/event/provisioning/service_order/*** event to the Provisioning Data Manager.

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

PCM_OP_PROV_UPDATE_SVC_ORDER

Updates the status of an **/event/provisioning/service_order/*** event.

An **/event/provisioning/service_order/*** event stores the service order and information such as the status, service order type, and actions required.

When a response is received from a provisioning platform, this opcode uses information in the input flist to update the status of an **/event/provisioning/service_order/*** event.

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

Payment FM Policy Opcodes

The opcodes listed in Table 1–64 manipulate A/R functions and collect payments from customers.

Header File

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

Opcode Index

Table 1–64 Payment FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PYMT_POL_APPLY_FEE	Provides the ability to preprocess, filter, and extend the information available in failed payment fee events. See the following discussions in *BRM Configuring and Collecting Payments*: ■ Storing additional information with payment fees ■ How payment fees are applied ■ Customizing payment fees	Recommended
PCM_OP_PYMT_POL_CHARGE	Maps payment status responses from the payment gateway to the BRM database. See the discussion on customizing payment failure reason code in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_POL_COLLECT	For a specific account, evaluates online collection results and specifies further action as needed. See the discussion on customizing how the results of credit-card transactions are processed in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_GRANT_INCENTIVE	Enriches the input flist by specifying attributes used by real-time rating to determine whether a payment incentive can be granted. Also provides additional fields that are recorded when creating the **/event/billing/incentive** object. See the discussion on customizing how to grant payment incentives in *BRM Configuring and Collecting Payments*.	Recommended

Table 1–64 (Cont.) Payment FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PYMT_POL_MBI_DISTRIBUTE	Contains the default payment distribution logic to distribute the submitted account-level payment to multiple bill units. See the discussion on allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_OVER_PAYMENT	Determines action if money received is more than the sum of the total due of all the open items selected. See the discussion on handling overpayments and underpayments in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_PRE_COLLECT	Performs policy checks before the charge or payment occurs. See the following discussions in *BRM Configuring and Running Billing*: ■ Setting the minimum amount to charge ■ Customizing the policy source file for soft descriptors	Recommended
PCM_OP_PYMT_POL_PROVISION_INCENTIVE	Determines the payment date that should be considered when provisioning incentives. See the discussion on customizing how to trigger payment incentives in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_PURCHASE_DEAL	Applies discounts to topped up account balances. See the discussion on offering discount incentives with top-ups in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_SPEC_COLLECT	For an account, specifies how much should be collected during registration. See the discussion on customizing whether to charge at registration in *BRM Managing Customers*.	Recommended
PCM_OP_PYMT_POL_SPEC_VALIDATE	Specifies whether the payment method should be validated during registration. See the discussion on customizing the account used for credit card validation in *BRM Managing Customers*.	Recommended
PCM_OP_PYMT_POL_SUSPEND_PAYMENT	Provides information that directs a payment marked for suspense to the payment suspense account. See the discussion on customizing payment guidance to suspense in *BRM Configuring and Collecting Payments*.	Recommended

Table 1–64 (Cont.) Payment FM Policy Opcodes

Opcode	Description	Use
PCM_OP_PYMT_POL_UNDER_PAYMENT	Determines action if money received is less than the sum of the total due of all the open items selected. See the discussion on handling overpayments and underpayments in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_VALID_VOUCHER	Interacts with voucher management systems such as Voucher Manager to validate vouchers. See the discussion on customizing voucher validation in *BRM Telco Integration*.	Recommended
PCM_OP_PYMT_POL_VALIDATE	Determines the success or failure of online validation results. See the discussion on changing how BRM handles Paymentech address validation return codes in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_POL_VALIDATE_PAYMENT	Validates payments to determine whether they can be successfully posted. Also determines whether a failed, unconfirmed payment needs reversal and whether write-off reversals should be performed. See the following discussions in *BRM Configuring and Collecting Payments*: ■ Customizing payment suspense validation ■ Configuring Payment Suspense Manager to allocate externally initiated payments by due amount ■ Allocating account-level payments to multiple bill units	Recommended

PCM_OP_PYMT_POL_APPLY_FEE

Allows customization of payment fees by preprocessing, filtering, and extending the information available in failed payment fee events.

This opcode also enhances the **/event/billing/fee/failed_payment** object by providing additional fields that will be recorded in the object.

This opcode is called by the PCM_OP_PYMT_APPLY_FEE standard opcode.

See the following discussions in *BRM Configuring and Collecting Payments*:

- Storing additional information with payment fees
- How payment fees are applied
- Customizing payment fees

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 this opcode to apply fees for failed credit card and direct debit transactions based on the reason for failure.

This opcode is called by the PCM_OP_PYMT_CHARGE standard opcode.

See the discussion on customizing payment failure reason codes in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_COLLECT

Processes the result of a credit card transaction for a specified account.

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.

This opcode is called by the PCM_OP_PYMT_COLLECT standard opcode after the credit card has been charged.

See the discussion on customizing how the results of credit-card transactions are processed in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_GRANT_INCENTIVE

Enriches the input flist for PCM_OP_PYMT_GRANT_INCENTIVE by specifying additional event attributes used by real-time rating to determine whether a payment incentive will be granted.

This opcode is called by the PCM_OP_PYMT_GRANT_INCENTIVE standard opcode.

See the discussion on customizing how to grant payment incentives in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_MBI_DISTRIBUTE

Contains the default payment distribution logic to distribute the submitted account-level payment to multiple bill units.

This opcode is called by the PCM_OP_PYMT_MBI_DISTRIBUTE standard opcode.

This opcode searches for all the open **/bill** objects for the given **/account** object, sorted by the bill due date.

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, this opcode provides bill-level distribution. So, BRM considers only the open bill items for payment distribution. However, you can update this opcode to return bill-unit-level payment distribution. If bill-unit-level 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.

See the discussion on allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_OVER_PAYMENT

Allocates overpayment of funds. By default, this opcode returns the amount overpaid on the output flist. Excess monies remains in the payment item until they are manually redistributed with Payment Tool.

This opcode is called by the PCM_OP_PYMT_SELECT_ITEMS standard opcode.

See the discussion on handling overpayments and underpayments in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_PRE_COLLECT

Checks a batch of charges and refunds for any amounts below minimums before charging and refunding customers.

This opcode is called by the PCM_OP_PYMT_COLLECT standard opcode.

You can change the minimum credit card charge amount by modifying the default minimum payment amount in this opcode.

You can also customize this opcode to retrieve soft descriptor information that enables you to display the name under which you do business (your DBA name), product name, and customer service number on your customer's checking account or credit card statement.

See the following discussions in *BRM Configuring and Running Billing*:

- Setting the minimum amount to charge
- Customizing the policy source file for soft descriptors

PCM_OP_PYMT_POL_PROVISION_INCENTIVE

Determines the payment date that should be considered when provisioning incentives. By default, this opcode reads the PIN_FLD_END_T field to obtain the timestamp.

You can customize this opcode 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 this opcode 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.

This opcode is called by the PCM_OP_PYMT_PROVISION_INCENTIVE standard opcode.

See the discussion on customizing how to trigger payment incentives in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_PURCHASE_DEAL

Applies custom discounts and incentives to account balances when an account is topped up.

This opcode is called by the PCM_OP_PYMT_TOPUP standard opcode.

See the discussion on offering discount incentives with top-ups in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_SPEC_COLLECT

Specifies how much should be collected from an account after a specified action has been performed. This opcode allows you to determine whether to charge the customer immediately for all or part of the current account balances during registration.

This opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER standard opcode.

See the discussion on customizing whether to charge at registration in *BRM Managing Customers*.

PCM_OP_PYMT_POL_SPEC_VALIDATE

Changes the account used for credit card validation.

When validating a credit card at registration, BRM needs an account to validate the card with. By default, this is the root account. You cannot store this information with each account because the credit card validation is done before the account is created.

This opcode is called by the PCM_OP_CUST_PREP_CUSTOMER standard opcode.

See the discussion on customizing the account used for credit card validation in *BRM Managing Customers*.

PCM_OP_PYMT_POL_SUSPEND_PAYMENT

Provides information that guides a payment marked for suspense to the payment suspense account.

Use this opcode to customize the process for guiding payments to suspense.

This opcode is called by the PCM_OP_PYMT_COLLECT standard opcode whenever it receives a payment that has the PIN_FLD_STATUS field set to PIN_PYMT_SUSPENSE. It checks the **/config/psm** object to determine the POID of the payment suspense account, and it returns all payment information in the output flist to PCM_OP_PYMT_COLLECT so BRM can post the payment to the payment suspense account as an unallocated payment.

See the discussion on customizing payment guidance to suspense in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_UNDER_PAYMENT

Allocates underpayment of funds.

By default, this opcode 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.

This opcode is called by the PCM_OP_PYMT_SELECT_ITEMS standard opcode.

See the discussion on handling overpayments and underpayments in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_VALID_VOUCHER

Interacts with voucher management systems such as Voucher Manager to validate vouchers.

This opcode is called by the PCM_OP_PYMT_TOPUP standard opcode during voucher top-up operations.

To interact with a voucher management system, this opcode calls the PCM_OP_VOUCHER_ASSOCIATE_VOUCHER standard opcode. See the discussion on performing top-ups with PCM_OP_PYMT_TOPUP in *BRM Configuring and Collecting Payments*.

To enable this opcode to work with a custom voucher management system, see the discussion on customizing voucher validation in *BRM Telco Integration*.

PCM_OP_PYMT_POL_VALIDATE

Returns the result of validating a credit card transaction, including a description of that result.

This opcode is called by the PCM_OP_PYMT_VALIDATE standard opcode.

See the discussion on changing how BRM handles Paymentech address validation return codes in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_POL_VALIDATE_PAYMENT

Validates payments to determine whether they can be successfully posted or whether a failed, unconfirmed payment needs reversal.

This opcode also identifies if the account-level 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.

This opcode is called by the PCM_OP_PYMT_VALIDATE_PAYMENT standard opcode.

See the following discussions:

- Customizing payment suspense validation in *BRM Configuring and Collecting Payments*
- Allocating externally initiated payments by due amount in *BRM Configuring and Collecting Payments*
- Allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*

Payment FM Standard Opcodes

The opcodes in Table 1–65 collect payments and validate payment methods.

Header File

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

Opcode Index

Table 1–65 Payment FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PYMT_APPLY_FEE	Records failed payments and applies payment fees. See the discussion on how payment fees are applied in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_CHARGE	Performs a BRM-initiated payment transaction. See the discussion on how BRM-initiated payment transactions are performed in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_CHARGE_CC	Performs an online credit card transaction. See the discussion on how BRM performs credit-card charges in *BRM Configuring and Collecting Payments*.	Last Resort
PCM_OP_PYMT_CHARGE_DD	Performs a Paymentech direct debit transaction. See the discussion on how BRM performs a batch of direct-debit charges in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_CHARGE_DDEBIT	Performs a direct debit card transaction. See the discussion on how BRM performs direct-debit charges in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_COLLECT	Performs payment collections and refunds. See the following discussions: ▪ Applying multiple payments to an account via payment gateways in *BRM Configuring and Collecting Payments* ▪ Allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*	Recommended

Table 1–65 (Cont.) Payment FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PYMT_FIND_TOPUP_EVENTS	Finds the **/event/billing/adjustment/account** events associated with sponsored top-ups. See the discussion on viewing sponsored top-up history in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_GET_ACH_INFO	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.	Recommended
PCM_OP_PYMT_GRANT_INCENTIVE	Applies a payment incentive to a bill during the billing run. See the discussion on how payment incentives work in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_ITEM_SEARCH	Searches the **/item** object with a variable number of input parameters. See the discussion on finding items in *BRM Managing Accounts Receivable*.	Limited
PCM_OP_PYMT_MBI_DISTRIBUTE	Distributes the account-level payment to multiple bill units. See the discussion on allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_MBI_ITEM_SEARCH	Retrieves the bills or item across multiple bill units of the account. See the discussion on allocating an account-level payment to multiple bill units in *BRM Configuring and Collecting Payments*.	Limited
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. See the discussion on how payment incentives are triggered in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_RECOVER	Checks results of charges sent in a batch. See the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_RECOVER_CC	Checks results of credit card charges sent in a batch. See the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.	Limited

Table 1–65 (Cont.) Payment FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PYMT_RECOVER_DD	Checks the results of direct debit charges sent in a batch. See the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_RECYCLE_PAYMENT	Removes a payment from the payment suspense account and posts it to the correct account. See the discussion on how payments are recycled to and from suspense in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH	During payment suspense processing, returns a list of distributed payments and retrieves recycled payment information for PCM_OP_PYMT_RECYCLE_PAYMENT for processing. See the discussion on how recycled payments are retrieved in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_REVERSE_INCENTIVE	Reverses a payment incentive, provided the incentive has not yet been applied. See the discussion on how payment incentives are reversed in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_SELECT_ITEMS	Identifies a list of items based on the input fields and the accounting type of the account. See the discussion on selecting the items to which payments are applied in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_TOPUP	Performs standard top-ups and sponsored top-ups. See the discussion on how BRM performs top-ups in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_PYMT_VALIDATE	Performs generic payment validations. See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.	Recommended

Table 1–65 (Cont.) Payment FM Standard Opcodes

Opcode	Description	Use
PCM_OP_PYMT_VALIDATE_CC	Performs a batch of credit card validations and applies the validation policy to the results. See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_VALIDATE_DD	Performs a batch of credit card validations and applies the validation policy to the results. See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.	Limited
PCM_OP_PYMT_VALIDATE_PAYMENT	Validates payments and prepares payments for posting by enriching the payment information. See the discussion on how payments are suspended during payment processing in *BRM Configuring and Collecting Payments*.	Recommended

PCM_OP_PYMT_APPLY_FEE

Creates payment fees for payments that fail; for example, due to insufficient account funds or an expired credit card.

This opcode calls PCM_OP_ACT_USAGE to create the payment fee event to be rated.

This opcode is called by PCM_OP_PYMT_COLLECT.

See the discussion on how payment fees are applied in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_CHARGE

Performs a BRM-initiated payment transaction.

This opcode is called by PCM_OP_PYMT_COLLECT, and is the recommended 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.

This opcode 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.

See the discussion on how BRM-initiated payment transactions are performed in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_CHARGE_CC

Performs an online credit card transaction. 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.

This opcode supports all commands handled by PCM_OP_PYMT_CHARGE.

See the discussion on how BRM performs credit-card charges in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_CHARGE_DD

Performs a batch of Paymentech direct debit transactions.

This opcode 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. See the "PCM_OP_PYMT_CHARGE" opcode for a list of the PIN result codes from BRM-initiated payment transactions.

See the discussion on how BRM performs a batch of direct-debit charges in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_CHARGE_DDEBIT

Performs a debit card transaction. This opcode is used for the Paymentech direct debit implementation shipped with BRM and used in creating a custom direct debit implementation for the bank or payment clearing house you choose.

Important: Debit cards that require a personal identification number (PIN) are not supported.

See the discussion on how BRM performs direct-debit charges in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_COLLECT

Performs payment collections and refunds.

This opcode allocates the payment to open items for each bill unit (**/billinfo** object) specified for the account. This opcode calls other standard opcodes to validate payments and calls various policy opcodes that allow you to customize payment collection.

See the discussion on how BRM collects payments in *BRM Configuring and Collecting Payments*.

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.

See the discussion on viewing sponsored top-up history in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_GET_ACH_INFO

Gets the Oracle 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.

PCM_OP_PYMT_GRANT_INCENTIVE

Applies a payment incentive to a bill during the billing run.

See the discussion on how payment incentives work in *BRM Configuring and Collecting 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, this opcode 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.

See the discussion on finding items in *BRM Managing Accounts Receivable*.

PCM_OP_PYMT_MBI_DISTRIBUTE

Distributes the account-level payment to multiple bill units.

This opcode is called by PCM_OP_PYMT_COLLECT or by Payment Tool.

This opcode calls the PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode.

See the discussion on allocating account-level payments to multiple bill units in *BRM Configuring and Collecting Payments*.

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 is called by Payment Tool only while manually allocating the payment.

This opcode calls PCM_OP_PYMT_ITEM_SEARCH.

See the discussion on allocating an account-level payment to multiple bill units in *BRM Configuring and Collecting Payments*.

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.

This opcode is called by PCM_OP_BILL_ITEM_TRANSFER immediately after payment allocation, provided BRM is configured for payment incentives. This opcode 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.

See the discussion on how payment incentives are triggered in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_RECOVER

Checks results of charges sent in a batch and posts results of charges for which no information was returned.

See the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_RECOVER_CC

Checks results of credit card charges sent in a batch and posts results of credit card charges for which no information was returned.

This opcode is specific to the Paymentech DM.

See the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_RECOVER_DD

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 the discussion on how BRM checks the results of BRM-initiated batch payment operations in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_RECYCLE_PAYMENT

Processes payment reversals during payment recycling and assigns action owner codes to suspended payments. This opcode is called by Payment Center when a single payment or a list of distributed payments is transferred between the payment suspense account and one or more customer accounts.

This opcode uses the source account referenced in the input flist's PIN_FLD_EVENT_OBJ field and the destination account POID in the PIN_FLD_ACCOUNT_OBJ field to determine 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. This opcode then uses the number of payments in the CHARGES array to determine whether the reversal is for a single payment or a list of distributed payments.

For account-level payment to multiple bill units, there can be more than one event generated for an individual payment. So, the output flist of this opcode shows all the payment events.

See the discussion on how payments are recycled to and from suspense in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH

Searches for recycled payments that have not been reversed, including those recycled to the payment suspense account.

This opcode is called by PCM_OP_PYMT_RECYCLE_PAYMENT and returns a list of distributed payments to PCM_OP_PYMT_RECYCLE_PAYMENT for processing. This opcode also returns recycled payment information such as the payment amount, transaction ID, subtransaction ID, and account number to PCM_OP_PYMT_RECYCLE_PAYMENT for processing.

See the discussion on how recycled payments are retrieved in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_REVERSE_INCENTIVE

Reverses a payment incentive, provided the incentive has not yet been applied.

See the discussion on how payment incentives are reversed in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_SELECT_ITEMS

Identifies a list of items based on the input fields and the accounting type of the account.

When an account-level payment is made to an account having multiple bill units, this opcode processes more than one bill unit to get the item-level distribution corresponding to each bill unit.

In case of overpayment to an account, this opcode contains more than two PIN_FLD_BILLINFO arrays for the default bill unit. This opcode does not perform an item-level 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-level distribution twice in two different PIN_FLD_BILLINFO arrays.

See the discussion on selecting the items to which payments are applied in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_TOPUP

Performs standard top-ups and sponsored top-ups.

See the discussion on how BRM performs top-ups in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_VALIDATE

Validates a credit card or direct debit transaction.

This opcode reads the **/config/payment** object to determine the transaction type and the opcode to call and then calls the appropriate opcode to validate the transaction.

This opcode also calls the PCM_OP_PYMT_POL_VALIDATE policy opcode to determine the success or failure of a BRM-initiated payment transaction validation.

See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_VALIDATE_CC

Performs a batch of online credit card validations and applies the validation policy to the results.

See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_VALIDATE_DD

Performs a batch of online direct debit validations and applies the validation policy to the results.

This opcode calls the appropriate DM to process validations and returns the results to the Internet.

See the discussion on how BRM validates credit card and direct debit transactions in *BRM Configuring and Collecting Payments*.

PCM_OP_PYMT_VALIDATE_PAYMENT

Validates payment records.

This opcode is called by PCM_OP_PYMT_COLLECT or by Payment Tool.

When this opcode 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.

See the discussion on how payments are suspended during payment processing in *BRM Configuring and Collecting Payments*.

RADIUS Manager FM Policy Opcodes

Use the opcodes listed in Table 1–66 to customize RADIUS Manager.

Header File

Include the **ops/term.h** header file in all applications that call these opcodes. For more information, see the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–66 RADIUS Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_TERM_POL_ACCOUNTING	Facilitates arbitrary storing of incoming RADIUS attributes. See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_POL_AUTHORIZE	Merges attributes from the NAS and a user's account into a list to be returned to the NAS. See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_POL_REVERSE_IP	Maps the IP address to the relevant account and service objects. See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.	Recommended

PCM_OP_TERM_POL_ACCOUNTING

Facilitates arbitrary storing of incoming RADIUS attributes.

This opcode can be customized to set the event type (such as **/event/session/dialup/ascend**) and extract extra fields from PIN_FLD_ARGS and PIN_FLD_INHERITED_INFO. You can then add these fields to an extended **/event/session/dialup** event.

This opcode is called by the PCM_OP_TERM_IP_DIALUP_START_ACCOUNTING and PCM_OP_TERM_IP_DIALUP_UPDATE_ACCOUNTING standard opcodes.

See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.

PCM_OP_TERM_POL_AUTHORIZE

Merges attributes from the NAS (Network Access Server) and a user's account into a list to be returned to the NAS.

This opcode is called by the PCM_OP_TERM_IP_DIALUP_AUTHORIZE standard opcode.

See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.

PCM_OP_TERM_POL_REVERSE_IP

Finds open event session for given IP address and returns user information based on that session.

Is called by relevant IP address applications to map an IP address to the relevant account and service objects. Use it to find the user of a given IP address.

This opcode is not called by any opcode.

See the discussion on customizing RADIUS Manager opcodes in *BRM RADIUS Manager*.

RADIUS Manager FM Standard Opcodes

The opcodes listed in Table 1–67 are used by the RADIUS Manager for authentication, authorization, and accounting.

Header File

Include the **ops/term.h** header file in all applications that call these opcodes. For more information, see the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–67 RADIUS Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_TERM_IP_DIALUP_ACCOUNTING_OFF	Records the end of accounting. See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_ACCOUNTING_ON	Enables the RADIUS Manager to tell BRM that it is ready for service. See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_AUTHENTICATE	Authenticates a user. See the discussion on how RADIUS Manager performs authentication and authorization in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_AUTHORIZE	Assembles information from the NAS and a user's account. See the discussion on how RADIUS Manager performs authentication and authorization in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_START_ ACCOUNTING	Records the start of a previously authenticated IP dialup session. See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_STOP_ ACCOUNTING	Closes out a previously started IP dialup session. See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.	Recommended
PCM_OP_TERM_IP_DIALUP_UPDATE_ ACCOUNTING	Updates a previously started IP dialup session. See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.	Recommended

PCM_OP_TERM_IP_DIALUP_ACCOUNTING_OFF

Records the end of accounting.

See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_ACCOUNTING_ON

Enables the RADIUS Manager to tell BRM that it is ready for service.

See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_AUTHENTICATE

Authenticates a user.

See the discussion on how RADIUS Manager performs authentication and authorization in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_AUTHORIZE

Assembles information from the NAS and a user's account.

See the discussion on how RADIUS Manager performs authentication and authorization in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_START_ACCOUNTING

Records the start of a previously authenticated IP dialup session.

See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_STOP_ACCOUNTING

Closes out a previously started IP dialup session.

See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.

PCM_OP_TERM_IP_DIALUP_UPDATE_ACCOUNTING

Updates a previously started IP dialup session.

See the discussion on how RADIUS Manager performs accounting in *BRM RADIUS Manager*.

Rating FM Policy Opcodes

The opcodes listed in Table 1–68 are called by Activity FM opcodes to calculate charges and taxes for an event.

Header File

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

Opcode Index

Table 1–68 Rating FM Policy Opcodes

Opcode	Description	Use
PCM_OP_RATE_POL_EVENT_ZONEMAP	Returns the zone map name and impact category for an event. See the discussion on getting zone maps and impact categories from the Pipeline Manager database in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_RATE_POL_GET_TAXCODE	Returns a list of all available taxcodes. See the discussion on retrieving a list of tax codes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_RATE_POL_GET_TAX_SUPPLIER	Returns a list of all available tax suppliers. See the discussion on retrieving a list of tax suppliers in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_RATE_POL_MAP_TAX_SUPPLIER	Returns the **tax_supplier** POID or the tax supplier information, such as the ship-from and ship-to addresses and the business location code, from the **tax_supplier_map** lookup table. See the discussion on retrieving tax supplier data in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_RATE_POL_POST_RATING	Modifies **/event** object information after it has been rated. See the discussion on modifying rated events in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_RATE_POL_PRE_RATING	Calls the PCM_OP_RATE_POL_PROCESS_ERAS opcode to retrieve the usage type of an event. See the discussion on rating an event based on extended rating attributes in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_RATE_POL_PROCESS_ERAS	Adds extended rating attribute (ERA) information to an event. See the discussion on rating an event based on extended rating attributes in *BRM Setting Up Pricing and Rating*.	Recommended

Table 1–68 (Cont.) Rating FM Policy Opcodes

Opcode	Description	Use
PCM_OP_RATE_POL_POST_TAX	Modifies tax data after tax calculation. See the discussion on modifying tax data after calculating taxes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_RATE_POL_PRE_TAX	Modifies tax data before tax calculation. See the discussion on modifying tax data before calculating taxes in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_RATE_POL_TAX_LOC	Returns the tax-related locations for an account. See the discussion on retrieving tax location data in *BRM Configuring and Running Billing*.	Last Resort

PCM_OP_RATE_POL_EVENT_ZONEMAP

Returns the zone map name and impact category for an event from the Pipeline Manager database.

You can customize this policy to add new event storable classes that you have created, if those event storable classes use a real-time zoning pipeline.

This opcode is not called by any opcode.

See the discussion on getting zone maps and impact categories from the Pipeline Manager database in *BRM Setting Up Pricing and Rating*.

PCM_OP_RATE_POL_GET_TAXCODE

Returns a list of all the taxcodes 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 taxcode information.

This opcode is not called by any opcode.

See the discussion on retrieving a list of tax codes in *BRM Configuring and Running Billing*.

PCM_OP_RATE_POL_GET_TAX_SUPPLIER

This opcode returns a list of tax suppliers from the **/profile/tax_supplier** object.

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.

This opcode is not called by any opcode.

See the discussion on retrieving a list of tax suppliers in *BRM Configuring and Running Billing*.

PCM_OP_RATE_POL_MAP_TAX_SUPPLIER

Provides tax supplier information from the tax supplier map file or from the products array of the account. You can customize this policy to change how a tax supplier is derived for a specific BRM event.

This opcode is called by the PCM_OP_BILL_TAX_EVENT standard opcode.

See the discussion on retrieving tax supplier data in *BRM Configuring and Running Billing*.

PCM_OP_RATE_POL_POST_RATING

Use the PCM_OP_RATE_POL_POST_RATING policy opcode 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 opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on modifying rated events in *BRM Setting Up Pricing and Rating*.

Caution: Before calling the PCM_OP_RATE_POL_POST_RATING opcode, the PCM_OP_ACT_USAGE opcode stores the balance of a rated event in a balance group cache rather than in the BRM database. If you customize the PCM_OP_RATE_POL_POST_RATING opcode to call a standard opcode to change the balance of the rated event directly in the database or in a different cache, the balance group cache from which the PCM_OP_ACT_USAGE opcode fetches the final balance impact of the event may not reflect the change made by the PCM_OP_RATE_POL_POST_RATING opcode.

PCM_OP_RATE_POL_PRE_RATING

Use this opcode to modify events before rating.

This opcode calls the PCM_OP_RATE_POL_PROCESSS_ERAS policy opcode to get the usage type of an event.

This opcode supports most-called-number discounts. When the event being rated is of the type **/event/billing/cycle/discount/mostcalled**, the opcode searches for usage events that match the event type and impact category specified in the **/profile/mostcalled** object that applies to the discount.

This opcode is called by the PCM_OP_RATE_AND_DISCOUNT_EVENT standard opcode.

See the discussion on discounts based on query values in *BRM Configuring Pipeline Rating and Discounting*.

PCM_OP_RATE_POL_PROCESS_ERAS

Adds extended rating attribute (ERA) information to an event.

This policy opcode calls the PCM_OP_RATE_GET_ERAS standard opcode to find the valid service-level and account-level 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.

This opcode returns the output to the PCM_OP_RATE_POL_PRE_RATING policy opcode.

This opcode is called by the PCM_OP_RATE_POL_PRE_RATING policy opcode.

PCM_OP_RATE_POL_POST_TAX

Use this opcode to modify data after taxes are calculated.

You can use this opcode to process the output flist that your tax calculation software returns after calculating taxes.

This opcode is called by the PCM_OP_RATE_TAX_CALC standard opcode.

See the discussion on modifying tax data after calculating taxes in *BRM Configuring and Running Billing*.

PCM_OP_RATE_POL_PRE_TAX

Use this opcode to modify data before you send the data to the taxation DM for calculating taxes.

By default, this opcode returns the input flist as the output flist.

In the default implementation, this opcode is called by the PCM_OP_RATE_TAX_CALC standard opcode before determining the tax package to use for tax calculation.

See the discussion on modifying tax data before calculating taxes in *BRM Configuring and Running Billing*.

PCM_OP_RATE_POL_TAX_LOC

Returns the locations for an event. These locations are then used to establish jurisdictions for tax calculation.

This opcode is called by the PCM_OP_RATE_EVENT standard opcode.

See the discussion on retrieving tax location data in *BRM Configuring and Running Billing*.

Rating FM Standard Opcodes

The opcodes listed in Table 1–69 are called by Activity FM opcodes to calculate charges and taxes for an event.

Header File

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

Opcode Index

Table 1–69 Rating FM Standard Opcodes

Opcode	Description	Use
PCM_OP_RATE_EVENT	Applies rates and taxes to an event. See the discussion on FM Rate opcodes called by PCM_OP_ACT_USAGE in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_RATE_GET_ERAS	Retrieves the extended rating attribute (ERAs) for an event. See the discussion on rating an event based on extended rating attributes in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_RATE_GET_PRODLIST	Retrieves the list of products owned by an account and filters them with input criteria. See the discussion on FM Rate opcodes called by PCM_OP_ACT_USAGE in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_RATE_TAX_CALC	Calculates taxes due. See the discussion on how BRM calculates taxes in *BRM Configuring and Running Billing*.	Last Resort
PCM_OP_RATE_TAX_EVENT	Directs tax calculation. See the discussion on how BRM calculates taxes in *BRM Configuring and Running Billing*.	Last Resort

PCM_OP_RATE_EVENT

Applies rates and taxes to an event. Uses the index value in the **/product** object's PIN_FLD_TAX_SUPPLIER field to locate the correct tax supplier in the cache.

When the optional timestamp field PIN_FLD_WHEN_T is present in the input flist, the opcode searches for the price list that is valid at the time specified in the field. It uses the price list to rate the event.

This opcode also locates any rollover objects for events, and returns details of the rollover object to the calling opcode.

See the discussion on FM Rate opcodes called by PCM_OP_ACT_USAGE in *BRM Setting Up Pricing and Rating*.

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.

PCM_OP_RATE_GET_PRODLIST

Retrieves the list of products owned by the account based on the combination of service and event type in the input flist. The list includes customized products that are currently valid as well as base products owned by the account.

See the discussion on FM Rate opcodes called by PCM_OP_ACT_USAGE in *BRM Setting Up Pricing and Rating*.

PCM_OP_RATE_TAX_CALC

Calculates taxes due at the time of purchase or billing.

This opcode is called by:

- PCM_OP_RATE_TAX_EVENT to calculate taxes on items that are taxable when purchased.
- PCM_OP_BILL_CYCLE_TAX to calculate taxes on items that are totalled and taxed at billing time.

See the discussion on how BRM calculates taxes in *BRM Configuring and Running Billing*.

PCM_OP_RATE_TAX_EVENT

Directs the calculation of taxes. This opcode is the main tax calculation opcode.

See the discussion on how BRM calculates taxes in *BRM Configuring and Running Billing*.

Remittance FM Policy Opcode

The opcode in Table 1–70 is used to retrieve the quantity to rate for a customized ratable usage metric (RUM).

Header File

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

Opcode Index

Table 1–70 Remittance FM Policy Opcode

Opcode	Description	Use
PCM_OP_REMIT_POL_SPEC_QTY	Retrieves the quantity to be rated for a customized RUM. See the discussion on customizing remittance in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_REMIT_POL_SPEC_QTY

Retrieves the quantity to rate for a customized ratable usage metric (RUM).

This opcode is called by the PCM_OP_REMIT_GET_PROVIDER standard opcode for events that use a custom RUM.

See the discussion on customizing remittance in *BRM Configuring and Running Billing*.

Remittance FM Standard Opcodes

The opcodes listed in Table 1–71 are used to manage remittance.

Header File

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

Opcode Index

Table 1–71 Remittance FM Standard Opcodes

Opcode	Description	Use
PCM_OP_REMIT_GET_PROVIDER	Retrieves a list of remittance accounts that need to be remitted when a particular event occurs. See the discussion on retrieving remittance accounts in *BRM Configuring and Running Billing*.	Limited
PCM_OP_REMIT_REMIT_PROVIDER	Calculates the remittance amount. See the discussion on calculating the remittance amount in *BRM Configuring and Running Billing*.	Limited
PCM_OP_REMIT_VALIDATE_SPEC_FLDS	Validates remittance criteria. See the discussion on verifying the remittance specification file in *BRM Configuring and Running Billing*.	Limited

PCM_OP_REMIT_GET_PROVIDER

Retrieves the list of remittance accounts that need to be remitted when a particular event occurs and then stores the information in **/remittance_info** objects. This data is later used by PCM_OP_REMIT_REMIT_PROVIDER to calculate the remittance amount owed to the service provider.

See the discussion on retrieving remittance accounts in *BRM Configuring and Running Billing*.

PCM_OP_REMIT_REMIT_PROVIDER

Calculates the remittance amount. This opcode is called directly by the **pin_remittance** utility.

See the discussion on calculating the remittance amount in *BRM Configuring and Running Billing*.

PCM_OP_REMIT_VALIDATE_SPEC_FLDS

Validates the **pin_remittance_spec** file. This opcode is called directly by the **load_pin_remittance_spec** utility.

See the discussion on verifying the remittance specification file in *BRM Configuring and Running Billing*.

Replication FM Policy Opcode

The opcode in Table 1–72 implements the translation logic for **/account** and **/service** objects.

Header File

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

Opcode Index

Table 1–72 Replication FM Policy Opcode

Opcode	Description	Use
PCM_OP_REPL_POL_PUSH	Implements the translation logic for **/account** and **/service** objects and publishes data to the LDAP database specified in PIN_FLD_CONSUMER_OBJ value the input flist. See the discussion on understanding the replication policy push operation in *BRM LDAP Manager*.	Recommended

PCM_OP_REPL_POL_PUSH

Implements 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.

This opcode is not called by any opcode.

See the discussion on understanding the replication policy push operation in *BRM LDAP Manager*.

For more information on the user mapping scheme, see the discussion on defining the user mapping scheme in *BRM LDAP Manager*.

Rerating FM Standard Opcode

The opcode in Table 1–73 calls other standard opcodes to create **/job/rerate** objects and **/job_batch/rerate** objects.

Header File

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

Opcode Index

Table 1–73 Rerating FM Standard Opcode

Opcode	Description	Use
PCM_OP_RERATE_INSERT_RERATE_REQUEST	Calls other standard opcodes to create **/job/rerate** objects and **/job_batch/rerate** objects. See the discussion on how BRM creates rerate jobs in *BRM Configuring and Running Billing*.	Recommended

PCM_OP_RERATE_INSERT_RERATE_REQUEST

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

See the discussion on how BRM creates rerate jobs in *BRM Configuring and Running Billing*.

Resource Reservation FM Policy Opcodes

Use the opcodes in Table 1–74 to customize processing of the reservation.

Header File

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

Opcode Index

Table 1–74 Resource Reservation FM Policy Opcodes

Opcode	Description	Use
PCM_OP_RESERVE_POL_POST_DISPUTE	Creates a reservation for a disputed amount to prevent misuse of resources during the course of the dispute. See the discussion on disputing events in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_RESERVE_POL_POST_SETTLEMENT	Releases the reservation against a disputed amount as part of the settlement process. See the discussion on settling disputed events in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_RESERVE_POL_PRE_RELEASE	Allows customization before releasing resources. See the discussion on customizing the rules for releasing a reservation in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_POL_PREP_CREATE	Determines the availability of resources for creating a reservation object. See the discussion on customizing resource reservation rules in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_POL_PREP_EXTEND	Determines the availability of resources for extending a reservation amount. See the discussion on customizing the rules for extending a reservation in *BRM Configuring and Collecting Payments*.	Recommended

PCM_OP_RESERVE_POL_POST_DISPUTE

Creates a reservation for a disputed amount to prevent misuse of resources during the dispute. PCM_OP_RESERVE_POL_POST_DISPUTE allows you to perform custom processing of the reservation that it creates for the dispute. BRM calls this opcode to reserve resources equivalent to the dispute amount for as long as the dispute is active.

This opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on disputing events in *BRM Managing Accounts Receivable*.

PCM_OP_RESERVE_POL_POST_SETTLEMENT

Releases the reservation against a disputed amount as part of the settlement process. PCM_OP_RESERVE_POL_POST_SETTLEMENT allows you to perform custom processing of the reservation that it releases during the settlement. BRM calls this opcode to release resources reserved by PCM_OP_RESERVE_POL_POST_DISPUTE.

This opcode is called by the PCM_OP_ACT_USAGE standard opcode.

See the discussion on settling disputed events in *BRM Managing Accounts Receivable*.

PCM_OP_RESERVE_POL_PRE_RELEASE

Specifies how to handle any unused resources when releasing a **/reservation** or **/reservation/active** object. By default, it is an empty hook provided to facilitate customization.

You can customize this opcode to offer customers the option of transferring unused resources to a different account.

This opcode is called by the PCM_OP_RESERVE_RELEASE standard opcode before it releases resources.

See the discussion on customizing the rules for releasing a reservation in *BRM Configuring and Collecting Payments*.

PCM_OP_RESERVE_POL_PREP_CREATE

Determines the availability of resources for creating a reservation.

You can customize this policy opcode to include custom resource reservation rules. For example, you can:

- Reserve whatever resource is available, even if the available resource is less than the requested amount.
- Specify a tolerance for credit limits.

This opcode is called by the PCM_OP_RESERVE_CREATE standard opcode before it creates a **/reservation** or **/reservation/active** object.

See the discussion on customizing resource reservation rules in *BRM Configuring and Collecting Payments*.

PCM_OP_RESERVE_POL_PREP_EXTEND

Determines the availability of resources for extending a reservation amount.

You can customize this policy opcode to include custom reservation rules. For example, you can:

- Extend the reserved amount even if the available resource is less than the requested amount.
- Extend reserved resources by specifying a resource limit and floor.

This opcode is called by the PCM_OP_RESERVE_EXTEND standard opcode before it extends the reservation amount.

See the discussion on customizing the rules for extending a reservation in *BRM Configuring and Collecting Payments*.

Resource Reservation FM Standard Opcodes

The opcodes in Table 1–75 manage reservations for prepaid services.

Header File

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

Opcode Index

Table 1–75 Resource Reservation FM Standard Opcodes

Opcode	Description	Use
PCM_OP_RESERVE_CREATE	Creates a **/reservation** or **/reservation/active** object. See the discussion on creating reservations in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_ASSOCIATE	Associates a session object with a reservation object. See the discussion on associating a session with a reservation in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_EXTEND	Extends a reservation amount. See the discussion on extending the reservation amount in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_FIND_OBJ	Finds one or more **/reservation** or **/reservation/active** objects. See the discussion on finding a reservation in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_RELEASE	Releases a **/reservation** or **/reservation/active** object. See the discussion on releasing reservations in *BRM Configuring and Collecting Payments*.	Recommended
PCM_OP_RESERVE_RENEW	Extends a reservation expiration time. See the discussion on extending the expiration time for a reservation in *BRM Configuring and Collecting Payments*.	Recommended

PCM_OP_RESERVE_CREATE

Creates a **/reservation** or **/reservation/active** object.

See the discussion on creating reservations in *BRM Configuring and Collecting Payments*.

If PIN_FLD_PIGGYBACK_FLAG is enabled in the input flist, this opcode returns information on the credit threshold breached in the PIN_FLD_PIGGYBACK_ NOTIFICATIONS array of the output flist.

See the discussion on providing credit threshold breach information in in-session notifications for network connectivity applications in *BRM Telco Integration*.

PCM_OP_RESERVE_ASSOCIATE

Associates an **/event/session** or **/active_session** object with a **/reservation** or **/reservation/active** object.

See the discussion on associating a session with a reservation in *BRM Configuring and Collecting Payments*.

PCM_OP_RESERVE_EXTEND

Extends the amount reserved in an existing **/reservation** or **/reservation/active** object.

See the discussion on extending the reservation amount in *BRM Configuring and Collecting Payments*.

If PIN_FLD_PIGGYBACK_FLAG is enabled in the input flist, this opcode returns the credit threshold breached in the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist.

See the discussion on providing credit threshold breach information in in-session notifications for network connectivity applications in *BRM Telco Integration*.

PCM_OP_RESERVE_FIND_OBJ

Finds one or more **/reservation** or **/reservation/active** objects.

See the discussion on finding a reservation in *BRM Configuring and Collecting Payments*.

PCM_OP_RESERVE_RELEASE

Releases one or more **/reservation** or **/reservation/active** objects. The opcode either keeps or deletes the reservation object and returns any unused resources back to the customer's account balance.

> **Note:** If a reservation object is not specified, this opcode searches for and releases all expired **/reservation** and **/reservation/active** objects.

See the discussion on releasing reservations in *BRM Configuring and Collecting Payments*.

PCM_OP_RESERVE_RENEW

Extends the expiration time of a **/reservation** or **/reservation/active** object.

This opcode fails when the reservation has already expired or is no longer active.

See the discussion on extending the expiration time for a reservation in *BRM Configuring and Collecting Payments*.

SDK FM Standard Opcodes

The opcodes listed in Table 1–76 add, delete, and modify data dictionary components, including opcode mapping, storable classes, and fields.

Header File

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

Opcode Index

Table 1–76 SDK FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SDK_DEL_FLD_SPECS	Deletes field specifications. See the discussion on deleting field specifications in *BRM Developer's Guide*.	Limited
PCM_OP_SDK_DEL_OBJ_SPECS	Deletes BRM storable class specifications. See the discussion on deleting storable class specifications in *BRM Developer's Guide*.	Limited
PCM_OP_SDK_GET_FLD_SPECS	Gets field specifications. See the discussion on retrieving field specifications in *BRM Developer's Guide*.	Limited
PCM_OP_SDK_GET_OBJ_SPECS	Gets BRM storable class specifications. See the discussion on retrieving storable class specifications in *BRM Developer's Guide*.	Limited
PCM_OP_SDK_SET_FLD_SPECS	Creates or modifies field specifications. See the discussion on creating and modifying field specifications in *BRM Developer's Guide*.	Limited
PCM_OP_SDK_SET_OBJ_SPECS	Creates or modifies BRM storable classes. See the discussion on creating and modifying storable classes in *BRM Developer's Guide*.	Limited

PCM_OP_SDK_DEL_FLD_SPECS

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.

See the discussion on deleting field specifications in *BRM Developer's Guide*.

Example 1–257 Sample Input Flist

This flist deletes a field named MY_FLD:

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /dd/fields 0 0
0 PIN_FLD_FIELD       ARRAY [0] allocated 1, used 1
1     PIN_FLD_FIELD_NAME      STR [0] "MY_FLD"
```

Example 1–258 Sample Output Flist

The POID of the deleted field is returned:

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /dd/fields 0 0
```

PCM_OP_SDK_DEL_OBJ_SPECS

Deletes storable class specifications from the data dictionary of all database schemas in your BRM system.

Note: The opcode deletes data from the data dictionary only. To drop the actual table created by PCM_OP_SDK_SET_OBJ_SPECS, you must drop it manually.

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.

See the discussion on deleting storable class specifications in *BRM Developer's Guide*.

Example 1–259 Sample Input Flist

This flist deletes a specification for a storable class of type **/my_class**:

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_OBJ_DESC      ARRAY [*] allocated 2, used 2
1     PIN_FLD_ACTION        STR [0] "delete"
1     PIN_FLD_NAME            STR [0] "/my_class"
```

Example 1–260 Sample Output Flist

This flist is the returned when a specification for a storable class of type **/my_class** is deleted:

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_RESULTS      ARRAY [0] allocated 1, used 1
1     PIN_FLD_DESCR      STR [0] "DELETE FROM dd_objects_t  WHERE
                    obj_id0 = 100064 "
0 PIN_FLD_RESULTS      ARRAY [1] allocated 1, used 1
1     PIN_FLD_DESCR      STR [0] "-- DROP TABLE my_class_t "
0 PIN_FLD_RESULTS      ARRAY [2] allocated 1, used 1
1     PIN_FLD_DESCR      STR [0] "DELETE FROM dd_objects_fields_t
                    WHERE obj_id0 = 100064 AND ( rec_id = 100065
                    OR (rec_id = 0 AND parent_element_id = 100065
                  ))"
```

PCM_OP_SDK_GET_FLD_SPECS

Retrieves one or more field specifications. You specify the field specifications to return on the input flist.

Important: If no fields are specified, this opcode returns all field specifications in the BRM database, which could take a long time.

See the discussion on retrieving field specifications in *BRM Developer's Guide*.

Example 1–261 Sample Input Flist

Specify the field to search for by using its name or ID number. This example uses the field name:

```
0 PIN_FLD_POID      POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_FIELD     ARRAY [0] allocated 1, used 1
1     PIN_FLD_FIELD_NAME      STR [0] "PIN_FLD_BILLINFO"
```

Example 1–262 Sample Output Flist

This flist is returned when specifications are retrieved for the PIN_FLD_BILLINFO field:

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /dd/fields 0 0
0 PIN_FLD_FIELD      ARRAY [0] allocated 5, used 5
1     PIN_FLD_DESCR           STR [0] NULL
1     PIN_FLD_FIELD_NAME      STR [0] "PIN_FLD_BILLINFO"
1     PIN_FLD_FIELD_NUM       ENUM [0] 126
1     PIN_FLD_FIELD_TYPE      INT [0] 9
1     PIN_FLD_STATUS             ENUM [0] 3
```

PCM_OP_SDK_GET_OBJ_SPECS

Retrieves one or more storable class specifications.

See the discussion on retrieving storable class specifications in *BRM Developer's Guide*.

Example 1–263 Sample Input Flist

This flist retrieves specifications for the **/event/batch** storable class:

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_OBJ_DESC      ARRAY [0] allocated 1, used 1
1     PIN_FLD_NAME            STR [0] "/event/batch"
```

Example 1–264 Sample Output Flist

This flist is returned when specifications are retrieved for the **/event/batch** storable class:

```
0 PIN_FLD_POID          POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_OBJ_DESC      ARRAY [1160] allocated 10, used 10
1     PIN_FLD_READ_ACCESS      STR [0] "BrandLineage"
1     PIN_FLD_WRITE_ACCESS     STR [0] "BrandLineage"
1     PIN_FLD_AUDIT_FLAG       INT [0] 0
1     PIN_FLD_AU_SM_INFO       STR [0] NULL
1     PIN_FLD_CREATE_ACCESS    STR [0] "Any"
1     PIN_FLD_DESCR               STR [0] "Abstract class to indicate batch load
session data"
1     PIN_FLD_LABEL      STR [0] NULL
1     PIN_FLD_NAME       STR [0] "/event/batch"
1     PIN_FLD_SM_INFO    STR [0] NULL
1     PIN_FLD_STATUS       ENUM [0] 3
```

PCM_OP_SDK_SET_FLD_SPECS

Creates or modifies field specifications.

Caution: If you change field specifications for fields that have been instantiated, you will corrupt your database.

Important: 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.

See the discussion on creating and modifying field specifications in *BRM Developer's Guide*.

Example 1–265 Sample Input Flist

This flist creates a field specification for a field named MY_INT_FIELD:

```
0 PIN_FLD_POID      POID [0] 0.0.0.1 /dd/fields 0 0
0 PIN_FLD_FIELD      ARRAY [0] allocated 4, used 4
1     PIN_FLD_DESCR          STR [0] "test field"
1     PIN_FLD_FIELD_NAME       STR [0] "MY_INT_FIELD"
1     PIN_FLD_FIELD_NUM        ENUM [0] 10005
1     PIN_FLD_FIELD_TYPE       INT [0] 5
```

PCM_OP_SDK_SET_OBJ_SPECS

Creates or modifies a storable class.

Caution: If you change a storable class after it has been instantiated and populated with data, you will corrupt your database.

Important: Instead of using this opcode, it is safer and more reliable to create or modify storable class specifications by using the Storable Class Editor in Developer Center.

See the discussion on creating and modifying storable classes in *BRM Developer's Guide*.

Example 1–266 Sample Input Flist

This flist creates specifications for a storable class of type **/my_class**:

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_OBJ_DESC     ARRAY [0] allocated 4, used 4
1     PIN_FLD_READ_ACCESS        STR [0] "BrandLineage"
1     PIN_FLD_WRITE_ACCESS       STR [0] "BrandLineage"
1     PIN_FLD_NAME               STR [0] "/my_class"
1     PIN_FLD_SM_ITEM_NAME          STR [0] "my_class_t"
```

Example 1–267 Sample Output Flist

This flist is returned when specifications are created for a class of type **/my_class**:

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /dd/objects 0 0
0 PIN_FLD_RESULTS     ARRAY [0] allocated 1, used 1
1     PIN_FLD_DESCR          STR [0] "INSERT INTO dd_objects_t (
                  obj_id0, name, mod_t, state, permission, label,
                descr, sm_info, seq_start, read_access, write_
                  access, create_access, audit_flag, au_sm_info )
                  VALUES ( 1522, '/my_class', 0, 1, 0, '', '',
                  '', 1, 'L', 'L', 'N', 0, '' )"
0 PIN_FLD_RESULTS      ARRAY [1] allocated 1, used 1
1     PIN_FLD_DESCR           STR [0] "CREATE TABLE my_class_t (
                  poid_DB int , poid_ID0 int , poid_TYPE varchar2
                  (255) , poid_REV int , created_t int , mod_t
                int , read_access varchar2 (1) , write_access
                  varchar2 (1)  ) "
0 PIN_FLD_RESULTS      ARRAY [2] allocated 1, used 1
1     PIN_FLD_DESCR            STR [0] "INSERT INTO dd_objects_
                   fields_t ( obj_id0, rec_id, parent_element_id,
                 field_name, field_type, state, permission,
                 length, encryptable, sm_info, label, descr,
                   field_order, auditable, sm_item_name ) VALUES
                   ( 1522, 1523, 0, 'PIN_FLD_MAIN', 11, 1, 1, 0,
                 0, '', '', '', 0.0, 0, 'my_class_t' )"
```

Services Framework AAA Manager FM Helper Opcodes

The opcodes listed in Table 1–77 perform service-specific functions.

About Helper Opcodes

Helper opcodes are called by Services Framework AAA Manager FM standard opcodes during any of the following stages in the standard opcode's execution:

- SEARCH_SESSION
- PREP_INPUT
- VALIDATE_LIFECYCLE
- ACC_ON_OFF_SEARCH
- POST_PROCESS

You can configure an opcode to call the helper opcodes by using the **load_aaa_config_opcodemap_tcf** utility. See the discussion on configuring Services Framework to call helper opcodes in *BRM Telco Integration*.

Error Handling

All opcodes check if ebuf is set before performing each step. If the ebuf is set, processing stops and the ebuf exception is passed to the caller.

Header File

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

Opcode Index

Table 1–77 Services Framework AAA Manager FM Helper Opcodes

Opcode	Description	Use
PCM_OP_TCF_AAA_ACCOUNTING_PREP_INPUT	Prepares service-specific input flists for activity events. See the discussion on preparing service-specific flists for activity events in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT	Prepares service-specific input flists for authorization. See the discussion on preparing service-specific flists for authorization in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL	Sends data about the current call to the policy opcode specified in the **/config/opcodemap/tcf** object and then tags the current call as either a continuation call or a normal call. See the discussion on how real-time rating detects dropped calls and continuation calls in *BRM Telco Integration*.	Recommended

Table 1–77 (Cont.) Services Framework AAA Manager FM Helper Opcodes

Opcode	Description	Use
PCM_OP_TCF_AAA_REAUTHORIZE_PREP_INPUT	Prepares service-specific input flists for reauthorization. See the discussion on preparing service-specific flists for reauthorization in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_SEARCH_SESSION	Builds search templates for finding service-specific **/active_session** and **/event/session** objects. See the discussion on building service-specific search templates in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_STOP_ACCOUNTING_PREP_INPUT	Prepares service-specific input flists for ending prepaid accounting sessions. See the discussion on preparing service-specific flists for ending accounting sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_UPDATE_ACCOUNTING_PREP_INPUT	Prepares service-specific input flists for updating prepaid accounting sessions. See the discussion on preparing service-specific flists for updating accounting sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE	During authorization of a request for a service that uses a custom service life cycle, applies business rules in the **/config/lifecycle_states** object associated with the service to validate the request. See the discussion on managing service life cycles in *BRM Managing Customers*.	Recommended

PCM_OP_TCF_AAA_ACCOUNTING_PREP_INPUT

Aggregates service-specific data and then creates an input flist for rating activity events.

See the discussion on preparing service-specific flists for activity events in *BRM Telco Integration*.

PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT

Aggregates service-specific data and then creates an flist for authorizing prepaid sessions.

See the discussion on preparing service-specific flists for authorization in *BRM Telco Integration*.

PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL

Sends data about the current call to the policy opcode specified in the **/config/opcodemap/tcf** object and then tags the current call as either a continuation call or a normal call.

The PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL helper opcode performs the following:

- Finds the dropped call ERA associated with the service.
- Searches through the existing **/active_session** objects in memory to find objects with the same service type and caller number as the current call. It sorts by PIN_FLD_CREATED_T, in descending order, all **/active_session** objects that meet the criteria and then finds the **/active_session** objects that match the termination cause specified in the service-specific **/config/aaa/gsm/***xxx* object.
- At the MATCH_CONTINUOUS_CALL processing stage, sends the current call, the dropped call, configuration information from the service-specific **/config/aaa/gsm/***xxx* object, billing information, a list of intermediate **/active_session** objects, and the dropped call ERA to the policy opcode specified in the **/config/opcodemap/tcf** object.
- Depending on the value returned by the policy opcode, tags the current call as either a normal call or a continuation call.
- Optionally deletes any redundant **/active_session** objects from memory. This includes **/active_session** objects that meet all of these criteria:
  - Have the same caller number as the current call
  - Have the same service type as the current call
  - Have any of the following:
    * A timestamp that exceeds the maximum duration specified in the dropped call ERA
    * A call counter that surpasses the maximum number of intermediate calls specified in the dropped call ERA
    * A billing cycle that is different than that of the current call

By default, this helper opcode is called by PCM_OP_TCF_AAA_AUTHORIZE and PCM_OP_TCF_AAA_STOP_ACCOUNTING at the TAG_SESSION processing stage.

See the discussion on how real-time rating detects dropped calls and continuation calls in *BRM Telco Integration*.

Example 1–268 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /active_
session/telco/gsm -1 0
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 116229 0
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
0 PIN_FLD_PROGRAM_NAME                     STR [0] "testnap"
0 PIN_FLD_ACTIVE_SESSION_ID                STR [0] "parag001_16"
0 PIN_FLD_START_T                       TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
0 PIN_FLD_RESERVATION_OBJ                 POID [0] 0.0.0.1 /reservation/active -1 0
0 PIN_FLD_DROPPED_CALL_QUANTITY        DECIMAL [0] 0
0 PIN_FLD_EXTENDED_INFO              SUBSTRUCT [0] allocated 20, used 2
```

```
1   PIN_FLD_TELCO_INFO              SUBSTRUCT [0] allocated 20, used 3
2      PIN_FLD_NETWORK_SESSION_ID         STR [0] "parag001_16"
2      PIN_FLD_CALLING_FROM               STR [0] "0049100050"
2      PIN_FLD_CALLED_TO                  STR [0] "0049100109"
1   PIN_FLD_GSM_INFO                SUBSTRUCT [0] allocated 20, used 1
2      PIN_FLD_DIRECTION                 ENUM [0] 0
0 PIN_FLD_RATING_INFO               SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_EVENT                   SUBSTRUCT [0] allocated 20, used 11
2      PIN_FLD_POID                      POID [0] 0.0.0.1 /event/session/telco/gsm
-1 0
2      PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 116229 0
2      PIN_FLD_SERVICE_OBJ               POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
2      PIN_FLD_PROGRAM_NAME               STR [0] "testnap"
2      PIN_FLD_ACTIVE_SESSION_ID          STR [0] "parag001_16"
2      PIN_FLD_START_T                  TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
2      PIN_FLD_RESERVATION_OBJ           POID [0] 0.0.0.1 /reservation/active -1 0
2      PIN_FLD_DROPPED_CALL_QUANTITY  DECIMAL [0] 0
2      PIN_FLD_TELCO_INFO            SUBSTRUCT [0] allocated 20, used 5
3         PIN_FLD_NETWORK_SESSION_ID      STR [0] "parag001_16"
3         PIN_FLD_CALLING_FROM            STR [0] "0049100050"
3         PIN_FLD_CALLED_TO               STR [0] "0049100109"
3         PIN_FLD_BYTES_UPLINK        DECIMAL [0] 0
3         PIN_FLD_BYTES_DOWNLINK      DECIMAL [0] 0
2      PIN_FLD_GSM_INFO              SUBSTRUCT [0] allocated 20, used 3
3         PIN_FLD_DIRECTION              ENUM [0] 0
3         PIN_FLD_BYTES_IN                INT [0] 0
3         PIN_FLD_BYTES_OUT               INT [0] 0
2      PIN_FLD_END_T                    TSTAMP [0] (1154606401) Thu Aug  3 05:00:01
2006
1   PIN_FLD_MIN_QUANTITY              DECIMAL [0] 0
0 PIN_FLD_CONFIG_INFO               SUBSTRUCT [0] allocated 20, used 17
1   PIN_FLD_POID                         POID [0] 0.0.0.1
/config/aaa/gsm/telephony 136864 0
1   PIN_FLD_CREATED_T                  TSTAMP [0] (1154524127) Wed Aug  2 06:08:47
2006
1   PIN_FLD_MOD_T                      TSTAMP [0] (1154523291) Wed Aug  2 05:54:51
2006
1   PIN_FLD_READ_ACCESS                  STR [0] "L"
1   PIN_FLD_WRITE_ACCESS                 STR [0] "L"
1   PIN_FLD_ACCOUNT_OBJ                 POID [0] 0.0.0.1 /account 1 0
1   PIN_FLD_DESCR                        STR [0] ""
1   PIN_FLD_HOSTNAME                     STR [0] "-"
1   PIN_FLD_NAME                         STR [0] "Telco AAA Params Configuration"
1   PIN_FLD_PROGRAM_NAME                 STR [0] "load_pin_telco_aaa_params"
1   PIN_FLD_VALUE                        STR [0] ""
1   PIN_FLD_VERSION                      STR [0] ""
1   PIN_FLD_TELCO_INFO              SUBSTRUCT [0] allocated 20, used 4
2      PIN_FLD_DELETED_FLAG               INT [0] 3
2      IN_FLD_DUPLICATE_CHECK_TYPE       ENUM [0] 1
2      PIN_FLD_EXPIRATION_T            TSTAMP [0] (0) <null>
2      PIN_FLD_SUBSESSION_MODE           ENUM [0] 2
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY  ARRAY [0] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE      ENUM [0] 4
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY  ARRAY [1] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE      ENUM [0] 5
1   PIN_FLD_RESERVATION_INFO            ARRAY [0] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
```

```
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 2
2      PIN_FLD_RUM_NAME                   STR [0] "Duration"
2      PIN_FLD_UNIT                      ENUM [0] 0
1   PIN_FLD_RESERVATION_INFO           ARRAY [1] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 1
2      PIN_FLD_RUM_NAME                   STR [0] "Amount"
2      PIN_FLD_UNIT                      ENUM [0] 0
0 PIN_FLD_OPCODE                          INT [0] 4002
```

Example 1–269 Sample Output Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /active_
session/telco/gsm -1 0
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 116229 0
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
0 PIN_FLD_PROGRAM_NAME                     STR [0] "testnap"
0 PIN_FLD_ACTIVE_SESSION_ID                STR [0] "parag001_16"
0 PIN_FLD_START_T                        TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
0 PIN_FLD_RESERVATION_OBJ                 POID [0] 0.0.0.1 /reservation/active -1 0
0 PIN_FLD_DROPPED_CALL_QUANTITY        DECIMAL [0] 0
0 PIN_FLD_EXTENDED_INFO              SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_TELCO_INFO               SUBSTRUCT [0] allocated 20, used 3
2      PIN_FLD_NETWORK_SESSION_ID         STR [0] "parag001_16"
2      PIN_FLD_CALLING_FROM               STR [0] "0049100050"
2      PIN_FLD_CALLED_TO                  STR [0] "0049100109"
1   PIN_FLD_GSM_INFO                 SUBSTRUCT [0] allocated 20, used 1
2      PIN_FLD_DIRECTION                 ENUM [0] 0
0 PIN_FLD_RATING_INFO                SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_EVENT                    SUBSTRUCT [0] allocated 20, used 11
2      PIN_FLD_POID                      POID [0] 0.0.0.1 /event/session/telco/gsm
-1 0
2      PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 116229 0
2      PIN_FLD_SERVICE_OBJ               POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
2      PIN_FLD_PROGRAM_NAME               STR [0] "testnap"
2      PIN_FLD_ACTIVE_SESSION_ID          STR [0] "parag001_16"
2      PIN_FLD_START_T                  TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
2      PIN_FLD_RESERVATION_OBJ           POID [0] 0.0.0.1 /reservation/active -1 0
2      PIN_FLD_DROPPED_CALL_QUANTITY  DECIMAL [0] 0
2      PIN_FLD_TELCO_INFO           SUBSTRUCT [0] allocated 20, used 5
3         PIN_FLD_NETWORK_SESSION_ID      STR [0] "parag001_16"
3         PIN_FLD_CALLING_FROM            STR [0] "0049100050"
3         PIN_FLD_CALLED_TO               STR [0] "0049100109"
3         PIN_FLD_BYTES_UPLINK        DECIMAL [0] 0
3         PIN_FLD_BYTES_DOWNLINK      DECIMAL [0] 0
2      PIN_FLD_GSM_INFO             SUBSTRUCT [0] allocated 20, used 3
3         PIN_FLD_DIRECTION               ENUM [0] 0
3         PIN_FLD_BYTES_IN                 INT [0] 0
3         PIN_FLD_BYTES_OUT                INT [0] 0
```

```
2      PIN_FLD_END_T                    TSTAMP [0] (1154606401) Thu Aug  3 05:00:01
2006
1   PIN_FLD_MIN_QUANTITY              DECIMAL [0] 0
0 PIN_FLD_CONFIG_INFO               SUBSTRUCT [0] allocated 20, used 17
1   PIN_FLD_POID                         POID [0] 0.0.0.1
/config/aaa/gsm/telephony 136864 0
1   PIN_FLD_CREATED_T                  TSTAMP [0] (1154524127) Wed Aug  2 06:08:47
2006
1   PIN_FLD_MOD_T                      TSTAMP [0] (1154523291) Wed Aug  2 05:54:51
2006
1   PIN_FLD_READ_ACCESS                   STR [0] "L"
1   PIN_FLD_WRITE_ACCESS                  STR [0] "L"
1   PIN_FLD_ACCOUNT_OBJ                  POID [0] 0.0.0.1 /account 1 0
1   PIN_FLD_DESCR                         STR [0] ""
1   PIN_FLD_HOSTNAME                      STR [0] "-"
1   PIN_FLD_NAME                          STR [0] "Telco AAA Params Configuration"
1   PIN_FLD_PROGRAM_NAME                  STR [0] "load_pin_telco_aaa_params"
1   PIN_FLD_VALUE                         STR [0] ""
1   PIN_FLD_VERSION                       STR [0] ""
1   PIN_FLD_TELCO_INFO               SUBSTRUCT [0] allocated 20, used 4
2      PIN_FLD_DELETED_FLAG               INT [0] 3
2      PIN_FLD_DUPLICATE_CHECK_TYPE      ENUM [0] 1
2      PIN_FLD_EXPIRATION_T            TSTAMP [0] (0) <null>
2      PIN_FLD_SUBSESSION_MODE           ENUM [0] 2
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY   ARRAY [0] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE       ENUM [0] 4
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY   ARRAY [1] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE       ENUM [0] 5
1   PIN_FLD_RESERVATION_INFO            ARRAY [0] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 2
2      PIN_FLD_RUM_NAME                   STR [0] "Duration"
2      PIN_FLD_UNIT                      ENUM [0] 0
1   PIN_FLD_RESERVATION_INFO            ARRAY [1] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 1
2      PIN_FLD_RUM_NAME                   STR [0] "Amount"
2      PIN_FLD_UNIT                      ENUM [0] 0
0 PIN_FLD_OPCODE                         INT [0] 4002
0 PIN_FLD_CALL_TYPE                      INT [0] 0
```

Return Values

The opcode returns the PIN_FLD_CALL_TYPE flist field set to one of the following to indicate whether the current call matches the criteria for a continuation call:

- **0** specifies this is a normal call.
- **1** specifies this is a dropped call.
- **2** specifies this is a continuation call.
- **3** specifies this is both a dropped call and a continuation call.

If the call is a continuation call, the opcode also returns the PIN_FLD_DROPPED_CALL_QUANTITY flist field set to the duration of the associated dropped call and the PIN_FLD_DROPPED_CALL_ASO_POID flist field set to the POID of the dropped call's **/active_session** object.

PCM_OP_TCF_AAA_REAUTHORIZE_PREP_INPUT

Aggregates service-specific data and then creates an flist for reauthorizing prepaid sessions.

See the discussion on preparing service-specific flists for reauthorization in *BRM Telco Integration*.

PCM_OP_TCF_AAA_SEARCH_SESSION

Builds search templates for finding **/active_session** and **/event/session** objects.

See the discussion on building service-specific search templates in *BRM Telco Integration*.

Example 1–270 Sample Input Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /active_session/telco
0 PIN_FLD_PROGRAM_NAME             STR [0] "sample_act"
0 PIN_FLD_AUTHORIZATION_ID         STR [0] "24874654"
0 PIN_FLD_DIRECTION               ENUM [0] 0
```

Example 1–271 Sample Output Flist

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_ARGS                   ARRAY [1] allocated 100, used 1
1   PIN_FLD_ACTIVE_SESSION_ID      STR [0] "00101771211133403461100004"
0 PIN_FLD_INDEX_NAME               STR [0] "active_session_active_id_i"
0 PIN_FLD_FLAGS                    INT [0] 256
0 PIN_FLD_TEMPLATE                 STR [0] "select X from /active_session/telco
where
                                           F1 = V1 "
0 PIN_FLD_RESULTS                ARRAY [0] NULL array ptr
```

PCM_OP_TCF_AAA_STOP_ACCOUNTING_PREP_INPUT

Aggregates service-specific data and create an flist for ending prepaid accounting sessions.

See the discussion on preparing service-specific flists for ending accounting sessions in *BRM Telco Integration*.

PCM_OP_TCF_AAA_UPDATE_ACCOUNTING_PREP_INPUT

Aggregates service-specific data and then creates an flist for a prepaid accounting sessions.

See the discussion on preparing service-specific flists for updating accounting sessions in *BRM Telco Integration*.

PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE

Does the following for all AAA requests except start accounting, update accounting, and stop accounting in normal mode:

1. Checks the **SubscriberLifeCycle** business parameter associated with a service's bill unit:
   - If the parameter is set to **disabled**, returns the authorization flow to the calling opcode, setting the **PIN_FLD_RESULT** output flist field to PIN_TCF_AAA_VALIDATION_NOT_REQUIRED.
   - If the parameter is set to **enabled**, continues the validation process.
2. Checks the PIN_FLD_LIFECYCLE_STATE value in the specified **/service** object:
   - If the value is **0**, returns the authorization flow to the calling opcode.
   - If the value is not **0**, continues the validation process.
3. Calls the PCM_OP_CUST_GET_LIFECYCLE_STATE opcode to get the life cycle states configuration object (**/config/lifecycle_states**) for the specified **/service** object.
4. Validates the call based on the business rules defined for the current state of the service.
5. Does one of the following:
   - If validation fails, sets the PIN_FLD_RESULT field in its output flist to PIN_BOOLEAN_FALSE and returns the authorization flow to the calling opcode. This triggers the calling opcode to return PIN_FLD_RESULT set to PIN_BOOLEAN_FALSE and PIN_FLD_REASON set to 7 (PIN_TCF_AAA_VALIDATION_FAILED) to its caller.
   - If validation succeeds, does one of the following:
     - If the current state of the service is Preactive or Dormant, allows the call and triggers a service state change by setting these output flist fields as follows:

       PIN_FLD_NEXT_STATE to ACTIVE

       PIN_FLD_RESULT to PIN_TCF_AAA_TRANSITION_REQUIRED
     - If the current state of the service is anything *except* Preactive or Dormant, sets the PIN_FLD_RESULT output flist field to PIN_BOOLEAN_TRUE.
6. Regardless of the validation results, calls the PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE policy opcode to complete the validation process.

This helper opcode is called during the VALIDATE_LIFECYCLE stage of the Services Framework AAA Manager FM standard opcodes. It can be called by any of those opcodes except the following:

- PCM_OP_TCF_AAA_START_ACCOUNTING
- PCM_OP_TCF_AAA_UPDATE_ACCOUNTING
- PCM_OP_TCF_AAA_STOP_ACCOUNTING in normal mode (when in direct debit mode, this standard opcode can call the helper opcode)

See the discussion on managing service life cycles in *BRM Managing Customers*.

Services Framework AAA Manager FM Policy Opcodes

The opcodes in Table 1–78 process authentication, authorization, and accounting (AAA) requests for any prepaid service type.

For more information about prepaid AAA, see the discussion on processing AAA requests for prepaid services in *BRM Telco Integration*.

Header File

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

Error Handling

All opcodes check whether ebuf is set before performing each step. If the ebuf is set, processing stops and the ebuf exception is passed to the caller.

Opcode Index

Table 1–78 Services Framework AAA Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE	Allows customizations when deploying BRM. See the discussion on customizing the information received by the policy controller in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL	Determines whether the current call is a continuation call. See the discussion on how real-time rating detects dropped calls and continuation calls in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_POL_POST_PROCESS	Sets up in-session notifications for use by network connectivity applications. See the discussion on providing in-session notifications in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE	During authorization of a request for a service that uses a custom life cycle, validates the request against the MO_ENABLED, MT_ENABLED, and custom business rules configured for the current service state. See the discussion on managing service life cycles in *BRM Managing Customers*.	Recommended

PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE

Customizes the information your network policy controller retrieves from BRM.

This policy opcode is called by the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode.

See the discussion on customizing the information received by the policy controller in *BRM Setting Up Pricing and Rating*.

PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL

Important: The source code for this policy opcode is not shipped with BRM. To modify how BRM finds continuation calls, you must create a custom policy opcode. See the discussion on specifying the rules for finding continuation calls in *BRM Telco Integration*.

Determines whether the current call is a continuation call by using the criteria specified in the dropped call extended rating attribute (ERA). The input to the policy opcode includes data about the current call, the dropped call, the dropped call ERA, the service-specific **/config/aaa/gsm** object, the billing cycle, and the list of intermediate **/active_session** objects.

See the discussion on how real-time rating detects dropped calls and continuation calls in *BRM Telco Integration*.

Example 1–272 Sample Input Flist

```
0 PIN_FLD_POID                            POID [0] 0.0.0.1 /active_
session/telco/gsm -1 0
0 PIN_FLD_ACCOUNT_OBJ                     POID [0] 0.0.0.1 /account 116229 0
0 PIN_FLD_SERVICE_OBJ                     POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
0 PIN_FLD_PROGRAM_NAME                     STR [0] "testnap"
0 PIN_FLD_ACTIVE_SESSION_ID                STR [0] "parag001_16"
0 PIN_FLD_START_T                       TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
0 PIN_FLD_RESERVATION_OBJ                 POID [0] 0.0.0.1 /reservation/active -1 0
0 PIN_FLD_DROPPED_CALL_QUANTITY        DECIMAL [0] 0
0 PIN_FLD_EXTENDED_INFO              SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_TELCO_INFO               SUBSTRUCT [0] allocated 20, used 3
2      PIN_FLD_NETWORK_SESSION_ID         STR [0] "parag001_16"
2      PIN_FLD_CALLING_FROM               STR [0] "0049100050"
2 PIN_FLD_CALLED_TO                       STR [0] "0049100109"
1   PIN_FLD_GSM_INFO                 SUBSTRUCT [0] allocated 20, used 1
2 PIN_FLD_DIRECTION                       ENUM [0] 0
0 PIN_FLD_RATING_INFO                SUBSTRUCT [0] allocated 20, used 2
1   PIN_FLD_EVENT                    SUBSTRUCT [0] allocated 20, used 11
2      PIN_FLD_POID                       POID [0] 0.0.0.1 /event/session/telco/gsm
-1 0
2      PIN_FLD_ACCOUNT_OBJ                POID [0] 0.0.0.1 /account 116229 0
2      PIN_FLD_SERVICE_OBJ                POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
2      PIN_FLD_PROGRAM_NAME                STR [0] "testnap"
2      PIN_FLD_ACTIVE_SESSION_ID           STR [0] "parag001_16"
2      PIN_FLD_START_T                  TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
2      PIN_FLD_RESERVATION_OBJ            POID [0] 0.0.0.1 /reservation/active -1 0
2      PIN_FLD_DROPPED_CALL_QUANTITY   DECIMAL [0] 0
2      PIN_FLD_TELCO_INFO             SUBSTRUCT [0] allocated 20, used 5
3         PIN_FLD_NETWORK_SESSION_ID      STR [0] "parag001_16"
3         PIN_FLD_CALLING_FROM            STR [0] "0049100050"
3         PIN_FLD_CALLED_TO               STR [0] "0049100109"
3         PIN_FLD_BYTES_UPLINK         DECIMAL [0] 0
3         PIN_FLD_BYTES_DOWNLINK       DECIMAL [0] 0
2      PIN_FLD_GSM_INFO               SUBSTRUCT [0] allocated 20, used 3
3         PIN_FLD_DIRECTION               ENUM [0] 0
```

```
3         PIN_FLD_BYTES_IN               INT [0] 0
3         PIN_FLD_BYTES_OUT              INT [0] 0
2      PIN_FLD_END_T                   TSTAMP [0] (1154606401) Thu Aug  3 05:00:01
2006
1   PIN_FLD_MIN_QUANTITY             DECIMAL [0] 0
0 PIN_FLD_CONFIG_INFO               SUBSTRUCT [0] allocated 20, used 17
1   PIN_FLD_POID                         POID [0] 0.0.0.1
/config/aaa/gsm/telephony 136864 0
1   PIN_FLD_CREATED_T                 TSTAMP [0] (1154524127) Wed Aug  2 06:08:47
2006
1   PIN_FLD_MOD_T                     TSTAMP [0] (1154523291) Wed Aug  2 05:54:51
2006
1   PIN_FLD_READ_ACCESS                  STR [0] "L"
1   PIN_FLD_WRITE_ACCESS                 STR [0] "L"
1   PIN_FLD_ACCOUNT_OBJ                 POID [0] 0.0.0.1 /account 1 0
1   PIN_FLD_DESCR                        STR [0] ""
1   PIN_FLD_HOSTNAME                     STR [0] "-"
1   PIN_FLD_NAME                         STR [0] "Telco AAA Params Configuration"
1   PIN_FLD_PROGRAM_NAME                 STR [0] "load_pin_telco_aaa_params"
1   PIN_FLD_VALUE                        STR [0] ""
1   PIN_FLD_VERSION                      STR [0] ""
1   PIN_FLD_TELCO_INFO              SUBSTRUCT [0] allocated 20, used 4
2      PIN_FLD_DELETED_FLAG              INT [0] 3
2      PIN_FLD_DUPLICATE_CHECK_TYPE     ENUM [0] 1
2      PIN_FLD_EXPIRATION_T           TSTAMP [0] (0) <null>
2      PIN_FLD_SUBSESSION_MODE          ENUM [0] 2
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY   ARRAY [0] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE       ENUM [0] 4
1   PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE_ARRAY   ARRAY [1] allocated 20, used 1
2      PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE       ENUM [0] 5
1   PIN_FLD_RESERVATION_INFO           ARRAY [0] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 2
2      PIN_FLD_RUM_NAME                   STR [0] "Duration"
2      PIN_FLD_UNIT                      ENUM [0] 0
1   PIN_FLD_RESERVATION_INFO           ARRAY [1] allocated 20, used 8
2      PIN_FLD_INCR_QUANTITY          DECIMAL [0] 50
2      PIN_FLD_IS_PRIMARY_RUM            ENUM [0] 0
2      PIN_FLD_MIN_QUANTITY           DECIMAL [0] 0
2      PIN_FLD_QUANTITY               DECIMAL [0] 50
2      PIN_FLD_RATIO                      INT [0] 0
2      PIN_FLD_REQ_MODE                  ENUM [0] 1
2      PIN_FLD_RUM_NAME                   STR [0] "Amount"
2      PIN_FLD_UNIT                      ENUM [0] 0
0 PIN_FLD_OPCODE                         INT [0] 4002
0 PIN_FLD_LAST_BILL_T                 TSTAMP [0] (1154516538) Wed Aug  2 04:02:18
2006
0 PIN_FLD_NEXT_BILL_T                 TSTAMP [0] (1157180400) Sat Sep  2 00:00:00
2006
0 PIN_FLD_PROFILE_INFO              SUBSTRUCT [0] allocated 20, used 7
1   PIN_FLD_PROFILE_NAME                 STR [0] "DROPPED_CALL"
1   PIN_FLD_POID                         POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
1   PIN_FLD_DATA_ARRAY                  ARRAY [0] allocated 20, used 4
2      PIN_FLD_NAME                      STR [0] "MAX_INTERVENING_CALLS"
2      PIN_FLD_VALID_FROM             TSTAMP [0] (1154516554) Wed Aug  2 04:02:34
```

```
2006
2      PIN_FLD_VALID_TO                TSTAMP [0] (1185993000) Wed Aug  1 11:30:00
2007
2      PIN_FLD_VALUE                      STR [0] "10"
1   PIN_FLD_DATA_ARRAY                 ARRAY [1] allocated 20, used 4
2      PIN_FLD_NAME                       STR [0] "SAME_CALLED_PARTY"
2      PIN_FLD_VALID_FROM              TSTAMP [0] (1154516554) Wed Aug  2 04:02:34
2006
2      PIN_FLD_VALID_TO                TSTAMP [0] (1185993000) Wed Aug  1 11:30:00
2007
2      PIN_FLD_VALUE                      STR [0] "1"
1   PIN_FLD_DATA_ARRAY                 ARRAY [2] allocated 20, used 4
2      PIN_FLD_NAME                       STR [0] "MAX_TIME_TO_CONTINUATION_CALL"
2      PIN_FLD_VALID_FROM              TSTAMP [0] (1154516554) Wed Aug  2 04:02:34
2006
2      PIN_FLD_VALID_TO                TSTAMP [0] (1185993000) Wed Aug  1 11:30:00
2007
2      PIN_FLD_VALUE                      STR [0] "600"
1   PIN_FLD_DATA_ARRAY                 ARRAY [4] allocated 20, used 4
2      PIN_FLD_NAME                       STR [0] ""
2      PIN_FLD_VALID_FROM              TSTAMP [0] (0) <null>
2      PIN_FLD_VALID_TO                TSTAMP [0] (0) <null>
2      PIN_FLD_VALUE                      STR [0] ""
1   PIN_FLD_EXTRATING                SUBSTRUCT [0] allocated 20, used 1
2      PIN_FLD_LABEL                      STR [0] ""
0 PIN_FLD_INTERMEDIATE_ASO_LIST     SUBSTRUCT [0] allocated 0, used 0
0 PIN_FLD_DROPPED_CALL_INFO         SUBSTRUCT [0] allocated 24, used 24
1   PIN_FLD_POID                         POID [0] 0.0.0.1 /active_
session/telco/gsm 136769 2
1   PIN_FLD_CREATED_T                TSTAMP [0] (1154523667) Wed Aug  2 06:01:07
2006
1   PIN_FLD_MOD_T                    TSTAMP [0] (1154523687) Wed Aug  2 06:01:27
2006
1   PIN_FLD_READ_ACCESS                 STR [0] "L"
1   PIN_FLD_WRITE_ACCESS                STR [0] "L"
1   PIN_FLD_ACCOUNT_OBJ                POID [0] 0.0.0.1 /account 116229 0
1   PIN_FLD_ACTIVE_SESSION_ID           STR [0] "parag001_13"
1   PIN_FLD_AMOUNT                   DECIMAL [0] -1
1   PIN_FLD_CALL_TYPE                   INT [0] 0
1   PIN_FLD_DROPPED_CALL_ASO_POID      POID [0] 0.0.0.0  0 0
1   PIN_FLD_DROPPED_CALL_QUANTITY    DECIMAL [0] 0
1   PIN_FLD_END_T                    TSTAMP [0] (1154606700) Thu Aug  3 05:05:00
2006
1   PIN_FLD_PROGRAM_NAME                STR [0] "testnap"
1   PIN_FLD_SERVICE_OBJ                POID [0] 0.0.0.1
/service/telco/gsm/telephony 1169 8
1   PIN_FLD_SESSION_ID                  INT [0] 0
1   PIN_FLD_SESSION_OBJ                POID [0] 0.0.0.1 /event/session/telco/gsm
28992567 0
1   PIN_FLD_SESSION_TYPE               ENUM [0] 0
1   PIN_FLD_START_T                  TSTAMP [0] (1154606400) Thu Aug  3 05:00:00
2006
1   PIN_FLD_STATUS                     ENUM [0] 5
1   PIN_FLD_TIMEZONE_ID                 STR [0] ""
1   PIN_FLD_USAGE_TYPE                  STR [0] ""
1   PIN_FLD_RESERVATION_LIST           ARRAY [0] allocated 20, used 1
2      PIN_FLD_RESERVATION_OBJ          POID [0] 0.0.0.1 /reservation/active
138817 0
1   PIN_FLD_TELCO_INFO               SUBSTRUCT [0] allocated 20, used 14
2      PIN_FLD_BYTES_DOWNLINK         DECIMAL [0] 0
```

```
2      PIN_FLD_BYTES_UPLINK           DECIMAL [0] 0
2      PIN_FLD_CALLED_TO                  STR [0] "0049100110"
2      PIN_FLD_CALLING_FROM               STR [0] "0049100050"
2      PIN_FLD_DESTINATION_NETWORK        STR [0] ""
2      PIN_FLD_NETWORK_SESSION_CORRELATION_ID STR [0] ""
2      PIN_FLD_NETWORK_SESSION_ID         STR [0] "parag001_13"
2      PIN_FLD_ORIGIN_NETWORK             STR [0] ""
2      PIN_FLD_PRIMARY_MSID               STR [0] ""
2      PIN_FLD_SECONDARY_MSID             STR [0] ""
2      PIN_FLD_SVC_CODE                   STR [0] ""
2      PIN_FLD_SVC_TYPE                   STR [0] ""
2      PIN_FLD_TERMINATE_CAUSE           ENUM [0] 4
2      PIN_FLD_USAGE_CLASS                STR [0] ""
1   PIN_FLD_GSM_INFO                SUBSTRUCT [0] allocated 20, used 14
2      PIN_FLD_BYTES_IN                   INT [0] 0
2      PIN_FLD_BYTES_OUT                  INT [0] 0
2      PIN_FLD_CALLED_NUM_MODIF_MARK     ENUM [0] 0
2      PIN_FLD_CELL_ID                    STR [0] ""
2      PIN_FLD_DESTINATION_SID            STR [0] ""
2      PIN_FLD_DIALED_NUMBER              STR [0] ""
2      PIN_FLD_DIRECTION                 ENUM [0] 0
2      PIN_FLD_IMEI                       STR [0] ""
2      PIN_FLD_LOC_AREA_CODE              STR [0] ""
2      PIN_FLD_NUMBER_OF_UNITS            INT [0] 0
2      PIN_FLD_ORIGIN_SID                 STR [0] ""
2      PIN_FLD_QOS_NEGOTIATED            ENUM [0] 0
2      PIN_FLD_QOS_REQUESTED             ENUM [0] 0
2      PIN_FLD_SUB_TRANS_ID               STR [0] ""
```

Example 1–273 Sample Output Flist

```
0 PIN_FLD_POID                           POID [0] 0.0.0.1 /active_
session/telco/gsm -1 0
0 PIN_FLD_RESULT                         ENUM [0] 0
```

Return Values

The opcode returns PIN_FLD_RESULT set to one of the following to indicate whether the current call matches the criteria for a continuation call:

- **0** specifies this is not a continuation call.
- **1** specifies this is a continuation call.
- **2** specifies this is not a continuation call because the maximum time duration or maximum number of intermediate calls between a dropped call and a continuation call was exceeded.

PCM_OP_TCF_AAA_POL_POST_PROCESS

Important: The source code for this policy opcode is not shipped with BRM. To modify how BRM finds continuation calls, you must create a custom policy opcode. See the discussion on providing in-session notifications in *BRM Telco Integration*.

The following opcodes call the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode as the last step in their process:

- PCM_OP_TCF_AAA_AUTHORIZE
- PCM_OP_TCF_AAA_REAUTHORIZE
- PCM_OP_TCF_AAA_STOP_ACCOUNTING
- PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE

The PCM_OP_TCF_AAA_POL_POST_PROCESS opcode calls PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES to retrieve the preferences for the subscriber associated with the authorization or reauthorization request.

The PCM_OP_TCF_AAA_POL_POST_PROCESS opcode places the subscriber's preference for language and channel of communication for notifications in a PIN_FLD_PIGGYBACK_NOTIFICATIONS array in its output flist. Additionally, it uses the data from the calling opcode in its input flist to set up notifications about the subscriber's current balances, credit thresholds breached, streaming usage summaries, subscription expirations, and impending tariff change indications in instances of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of its output flist.

See the discussion on providing in-session notifications in *BRM Telco Integration*.

PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE

By default, does the following during authorization of a telco service request:

1. Does one of the following:
   - If the REQ_ALLOWED rule validation performed by the calling opcode passed, allows or rejects mobile-originated (MO) and mobile-terminated (MT) calls based on the MO_ENABLED and MT_ENABLED rules in the **/config/lifecycle_states** object associated with the service type.
   - If the REQ_ALLOWED rule validation performed by the calling opcode failed, goes to the next step.
2. Applies any custom business rules configured in this policy opcode for the current service state.
3. Returns the final results of the validation:
   - PIN_BOOLEAN_TRUE (validation succeeded)
   - PIN_BOOLEAN_FALSE (validation failed)
   - PIN_TCF_AAA_TRANSITION_REQUIRED plus the state to change to (PIN_FLD_NEXT_STATE)
   - PIN_TCF_AAA_VALIDATION_NOT_REQUIRED (custom service life cycles are not enabled)

This policy opcode is called by the PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE helper opcode, whose preliminary validation results it can override.

See the discussion on managing service life cycles in *BRM Managing Customers*.

Services Framework AAA Manager FM Standard Opcodes

The opcodes listed in Table 1–79 process AAA requests for any prepaid service type.

For more information about prepaid AAA, see the discussion on processing AAA requests for prepaid services in *BRM Telco Integration*.

Header File

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

Error Handling

All opcodes check if ebuf is set before performing each step. If the ebuf is set, processing stops and the ebuf exception is passed to the caller.

Helper Opcodes

Services Framework AAA Manager standard opcodes can call helper opcodes during any of these stages in the opcode's execution:

- PREP_INPUT
- SEARCH_SESSION
- VALIDATE_LIFECYCLE
- ACC_ON_OFF_SEARCH
- TAG_SESSION
- POST_PROCESS

Services Framework AAA opcodes call helper opcodes at these stages if you configure them to do so with the **load_aaa_config_opcodemap_tcf** utility.

See the discussion on configuring Services Framework to call helper opcodes in *BRM Telco Integration*.

Opcode Index

Table 1–79 Services Framework AAA Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_TCF_AAA_ACCOUNTING	Performs accounting for activity-based usage. See the discussion on rating and recording activity events in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_ACCOUNTING_OFF	Closes all open sessions when the network is being shut down or encounters a problem. See the discussion on closing prepaid sessions when the external network shuts down in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_ACCOUNTING_ON	Closes all open sessions when the network restarts. See the discussion on closing prepaid sessions when the external network restarts in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_AUTHENTICATE	Authenticates users for prepaid services. See the discussion on authenticating users for custom services in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_AUTHORIZE	Authorizes users to access prepaid services. See the discussion on authorizing prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION	Cancels an authorization and releases any reserved resources. See the discussion on canceling authorization for prepaid services in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE	Returns the static information or dynamic information (or both) for all the services associated with the service. See the discussion on the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_TCF_AAA_QUERY_BALANCE	Provides account balance information.	Recommended
PCM_OP_TCF_AAA_REAUTHORIZE	Reauthorizes prepaid sessions. See the discussion on reauthorizing prepaid sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_REFUND	Refunds charges for prepaid sessions. See the discussion on refunding charges for prepaid sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_SERVICE_PRICE_ENQUIRY	Provides service cost information.	Recommended

Table 1–79 (Cont.) Services Framework AAA Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_TCF_AAA_START_ACCOUNTING	Starts accounting sessions. See the discussion on starting prepaid sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_STOP_ACCOUNTING	Stops accounting sessions. See the discussion on ending prepaid sessions in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_UPDATE_ACCOUNTING	Updates information about an active prepaid session. See the discussion on updating a prepaid session in *BRM Telco Integration*.	Recommended
PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE	Reauthorizes prepaid sessions and updates the active session. See the discussion on updating and reauthorizing prepaid sessions in *BRM Telco Integration*.	Recommended

PCM_OP_TCF_AAA_ACCOUNTING

Performs accounting for activity-based usage.

See the discussion on rating and recording activity events in *BRM Telco Integration*.

Example 1–274 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
1254125
0 PIN_FLD_PROGRAM_NAME          STR [0] "sample_act
0 PIN_FLD_END_T              TSTAMP [0] (1116453137) Wed May 18 14:52:17 2005
0 PIN_FLD_STATUS               ENUM [0] 0
0 PIN_FLD_MSID                  STR [0] "17985551234"
```

Example 1–275 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
1254125
0 PIN_FLD_TOTAL               ARRAY [0] allocated 1, used 1
0   PIN_FLD_AMOUNT           DECIMAL [0] 20.5
```

PCM_OP_TCF_AAA_ACCOUNTING_OFF

Closes any open sessions for a specified service type when the external network shuts down or encounters a severe problem.

See the discussion on closing prepaid sessions when the external network shuts down in *BRM Telco Integration*.

Example 1–276 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
231654 10
0 PIN_FLD_ORIGIN_NETWORK        STR [0] "Sample Network"
0 PIN_FLD_ACC_FLAG              INT [0] 0
0 PIN_FLD_START_T             TSTAMP [0] (1095383091) Thu Sep 16 18:04:51 2004
```

Example 1–277 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
231654 10
```

PCM_OP_TCF_AAA_ACCOUNTING_ON

Closes any open sessions for a specified service type when the external network restarts.

See the discussion on closing prepaid sessions when the external network restarts in *BRM Telco Integration*.

Example 1–278 Sample Input Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
231654 10
0 PIN_FLD_ORIGIN_NETWORK        STR [0] "Sample Network"
0 PIN_FLD_ACC_FLAG              INT [0] 0
0 PIN_FLD_START_T            TSTAMP [0] (1095383091) Thu Sep 16 18:04:51 2004
```

Example 1–279 Sample Output Flist

```
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /service/telco/custom_service
231654 10
```

PCM_OP_TCF_AAA_AUTHENTICATE

Authenticates users for prepaid services.

See the discussion on authenticating users for custom services in *BRM Telco Integration*.

Example 1–280 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/data 021454 10
0 PIN_FLD_MSID            STR [0] "24095830"
0 PIN_FLD_PASSWD_CLEAR    STR [0] "     "
```

Example 1–281 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/data 021454 10
0 PIN_FLD_MSID            STR [0] "24095830"
0 PIN_FLD_PASSWORD        STR [0] "     "
0 PIN_FLD_REASON         ENUM [0] 1
0 PIN_FLD_RESULT         ENUM [0] 1
```

PCM_OP_TCF_AAA_AUTHORIZE

Authorizes customers to access prepaid services.

See the discussion on authorizing prepaid services in *BRM Telco Integration*.

If you enabled in-session notifications, this opcode also returns the following:

- Details on the call denial
- Notifications about the subscriber's current balances, credit thresholds breached, streaming usage summaries, subscription expiration, impending tariff change indication, and the subscriber's preferences for receiving such notifications in instances of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of its output flist

See the discussion on providing in-session notifications for network connectivity applications in *BRM Telco Integration*.

PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION

Cancels an authorization and returns reserved resources back to the customer's account balance.

See the discussion on canceling authorization for prepaid services in *BRM Telco Integration*.

PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE

Returns the static information or dynamic information (or both) for all the services associated with the account.

If the service POID is not passed in the input flist, this opcode calls the PCM_OP_ACT_FIND opcode to get all the services associated with the subscriber's mobile station ID (MSID) passed in the input flist.

See the discussion on the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode in *BRM Setting Up Pricing and Rating*.

PCM_OP_TCF_AAA_QUERY_BALANCE

Provides the account balance information.

See the discussion on requesting an account's balance information in *BRM Telco Integration*.

PCM_OP_TCF_AAA_REAUTHORIZE

Reauthorizes prepaid sessions, so customers can continue an existing session.

See the discussion on reauthorizing prepaid sessions in *BRM Telco Integration*.

If you enabled in-session notifications, this opcode also returns the following:

- Details on the call denial
- Notifications about the subscriber's current balances, credit thresholds breached, streaming usage summaries, subscription expiration, impending tariff change indication, and the subscriber's preferences for receiving such notifications in instances of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist

See the discussion on providing in-session notifications for network connectivity applications in *BRM Telco Integration*.

PCM_OP_TCF_AAA_REFUND

Refunds charges for prepaid sessions.

See the discussion on refunding charges for prepaid sessions in *BRM Telco Integration*.

PCM_OP_TCF_AAA_SERVICE_PRICE_ENQUIRY

Provides the cost information for a specific service.

See the discussion on requesting service price information in *BRM Telco Integration*.

PCM_OP_TCF_AAA_START_ACCOUNTING

Starts accounting sessions for a specified prepaid service type.

See the discussion on starting prepaid sessions in *BRM Telco Integration*.

PCM_OP_TCF_AAA_STOP_ACCOUNTING

Ends prepaid accounting sessions.

See the discussion on ending prepaid sessions in *BRM Telco Integration*.

If you enabled in-session notifications, this opcode also returns the following:

- Details on the call denial
- Notifications about the subscriber's current balances, credit thresholds breached, streaming usage summaries, subscription expiration, impending tariff change indication, and the subscriber's preferences for receiving such notifications in instances of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist

See the discussion on providing in-session notifications for network connectivity applications in *BRM Telco Integration*.

PCM_OP_TCF_AAA_UPDATE_ACCOUNTING

Updates information about an existing prepaid session.

See the discussion on updating a prepaid session in *BRM Telco Integration*.

PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE

Updates information about an existing prepaid session and then reauthorizes the session.

See the discussion on updating and reauthorizing prepaid sessions in *BRM Telco Integration*.

If you enabled in-session notifications, this opcode also returns the following:

- Details on the call denial
- Notifications about the subscriber's current balances, credit thresholds breached, streaming usage summaries, subscription expiration, impending tariff change indication, and the subscriber's preferences for receiving such notifications in instances of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array of the output flist

See the discussion on providing in-session notifications for network connectivity applications in *BRM Telco Integration*.

Services Framework Manager FM Policy Opcodes

The opcodes in Table 1–80 process AAA requests for any prepaid service type.

Header File

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

Opcode Index

Table 1–80 Services Framework Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_TCF_POL_APPLY_PARAMETER	Adds custom information to the service object.	Recommended
PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER	Performs custom actions to a provisioning service order	Recommended

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.

This opcode is called by the PCM_OP_TCF_APPLY_PARAMETER standard opcode.

Customizing the Opcode

By default, this policy opcode returns an empty inherited info flist.

Customization Example

If you added a substruct to a customized service, you can use this opcode to fill in the substruct fields. These fields will be 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 this opcode to add the bearer information through the service extension to update the service flist.

PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER

Performs custom actions to a provisioning service order before it is passed to **dm_prov_telco**.

This opcode is called by the PCM_OP_TCF_PROV_HANDLE_SVC_ORDER standard opcode.

By default, this policy opcode does nothing, but you can customize it to override the provisioning mode and modify service order event details.

See the discussion on customizing the provisioning mode based on service order attributes in *BRM Telco Integration*.

Services Framework Manager FM Provisioning Opcodes

The opcodes listed in Table 1–81 perform provisioning functions.

Header File

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

Opcode Index

Table 1–81 Services Framework Manager FM Provisioning Opcodes

Opcode	Description	Use
PCM_OP_TCF_APPLY_PARAMETER	Updates the objects impacted by the product provisioning update.	Recommended
PCM_OP_TCF_SVC_LISTENER	Determines whether the product provisioning update is deferred for the future.	Recommended
PCM_OP_TCF_PROPAGATE_STATUS	Propagates the service status change to associated features and ERAs.	Recommended
PCM_OP_TCF_PROV_CREATE_SVC_ORDER	Creates service orders for provisioning prepaid services, devices, and profiles.	Recommended
PCM_OP_TCF_PROV_HANDLE_SVC_ORDER	Prepares provisioning event data for publishing to the provisioning DM and initiates status change to PROVISIONING.	Recommended
PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT	Updates the status of the **/service/device** object upon receiving the response from the provisioning platform.	Recommended
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.	Recommended
PCM_OP_TCF_PROV_SERVICE_ORDER_NOTIFY	Handles service order state changes.	Recommended
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_ATTR	Updates a service order object.	Recommended
PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE	Sets the state for service order objects and validates the service order state transition using information stored in the configuration object.	Recommended

PCM_OP_TCF_APPLY_PARAMETER

This opcode updates the objects impacted by the product provisioning update. This opcode is called either by PCM_OP_TCF_SVC_LISTENER or by the schedule framework.

See the discussion on associating APNs and QoS with GPRS services in *BRM Telco Integration*.

PCM_OP_TCF_SVC_LISTENER

This opcode checks the action's start and end date and performs one of the following:

- If it is a current action, the opcode calls PCM_OP_TCF_APPLY_PARAMETER.
- If the action is deferred for the future, the opcode creates a **/schedule** object for executing the PCM_OP_TCF_APPLY_PARAMETER at the scheduled time.

This opcode is called by the event notification system.

See the discussion on associating APNs and QoS with GPRS services in *BRM Telco Integration*.

PCM_OP_TCF_PROPAGATE_STATUS

When the service status changes to inactive, active, or closed, the opcode propagates the status to any associated features and extended rating attributes (ERAs). When the status changes to closed, the opcode also disassociates any existing devices from the service.

This opcode is called by the event notification system when a service changes status.

PCM_OP_TCF_PROV_CREATE_SVC_ORDER

This opcode creates service orders for provisioning prepaid services, devices, and profiles.

PCM_OP_TCF_PROV_HANDLE_SVC_ORDER

Prepares provisioning event data for publishing to the provisioning DM and initiates status change to PROVISIONING.

PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT

Updates the status of the **/service/device** object upon receiving the response from the provisioning platform.

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.

PCM_OP_TCF_PROV_SERVICE_ORDER_NOTIFY

Handles service order state changes.

PCM_OP_TCF_PROV_SERVICE_ORDER_SET_ATTR

Updates the service order object.

PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE

Sets the state for service order objects and validates the service order state transition using information stored in the configuration object.

SIM Manager FM Policy Opcodes

Use the opcodes in Table 1–82 to customize SIM Manager.

Header File

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

Opcode Index

Table 1–82 SIM Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_SIM_POL_DEVICE_ASSOCIATE	Associates a device with a service disassociates a service from the device. See the discussion on customizing SIM card service association in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_POL_DEVICE_CREATE	During device creation, validates the SIM card number, IMSI, KI, and network element values. Also checks for duplicate SIM cards. See the discussion on customizing SIM card validation in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_POL_DEVICE_SET_ATTR	During device update, ensures that the SIM card number (PIN_FLD_DEVICE_ID) cannot be changed. See the discussion on customizing SIM card number changes in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_POL_DEVICE_SET_BRAND	When changing a SIM card brand, validates that the device state is Released. See the discussion on managing SIM cards in a branded system in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_POL_ORDER_CREATE	When you create a SIM order, validates the SIM order attributes.	Recommended

PCM_OP_SIM_POL_DEVICE_ASSOCIATE

Associates and disassociates a service with a device. You can customize this opcode to change how SIM cards and services are associated.

This opcode is called by the PCM_OP_DEVICE_POL_ASSOCIATE policy opcode.

See the discussion on customizing SIM card service association in *BRM Telco Integration*.

PCM_OP_SIM_POL_DEVICE_CREATE

This opcode validates a device by validating the SIM card number, IMSI, KI, and network element values. You can customize this opcode to change validation rules for creating SIM card devices.

This opcode is called by the PCM_OP_DEVICE_POL_CREATE policy opcode.

See the discussion on customizing SIM card validation in *BRM Telco Integration*.

PCM_OP_SIM_POL_DEVICE_SET_ATTR

Ensures that the SIM card number (PIN_FLD_DEVICE_ID) cannot be changed. You can customize this opcode to change how SIM cards are associated with services.

This opcode is called by the PCM_OP_DEVICE_POL_SET_ATTR policy opcode when updating a SIM card device.

See the discussion on customizing SIM card number changes in *BRM Telco Integration*.

PCM_OP_SIM_POL_DEVICE_SET_BRAND

When changing the SIM card brand, validates that the SIM card device state is Released. You can customize this opcode to change how SIM cards can be associated with brands.

This opcode is called by the PCM_OP_DEVICE_POL_SET_BRAND policy opcode.

See the discussion on managing SIM cards in a branded system in *BRM Telco Integration*.

PCM_OP_SIM_POL_ORDER_CREATE

When you create a SIM order, validates the SIM order attributes.

SIM Manager FM Standard Opcodes

The opcodes listed in Table 1–83 are used to create and manage SIM card objects in the BRM database.

Header File

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

Opcode Index

Table 1–83 SIM Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SIM_CREATE_ORDER	Creates a SIM card order object in the BRM database. See the discussion on creating and updating SIM card orders in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_DEVICE_PROVISION	Moves the device state from New to Provisioning, associates a service, and disassociates the pre-provisioning service. See the discussion on provisioning SIM cards in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_PROCESS_ORDER_RESPONSE	Processes a vendor response file. See the discussion on creating SIM Cards in *BRM Telco Integration*.	Recommended
PCM_OP_SIM_UPDATE_ORDER	Updates an existing order object. See the discussion on creating and updating SIM card orders in *BRM Telco Integration*.	Recommended

PCM_OP_SIM_CREATE_ORDER

Creates a SIM card order object (**/order/sim**).

Checks for duplicate SIM card and IMSI numbers. If no error is found, creates a SIM card order object and sets the order status to New.

See the discussion on creating and updating SIM card orders in *BRM Telco Integration*.

PCM_OP_SIM_DEVICE_PROVISION

Associates a SIM card with a service, and disassociates the pre-provisioning service.

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

PCM_OP_SIM_PROCESS_ORDER_RESPONSE

Processes a vendor response file.

The vendor response file includes a list of SIM cards that you load into the BRM database by using SIM Administration Center.

See the discussion on creating SIM Cards in *BRM Telco Integration*.

PCM_OP_SIM_UPDATE_ORDER

Updates an existing SIM card order object (**/order/sim**).

This opcode is called when a customer updates the order, or when the order status needs to be changed, for example, after processing a vendor response file. This opcode is also called when an order is canceled.

See the discussion on creating and updating SIM card orders in *BRM Telco Integration*.

Subscription Management FM Policy Opcodes

Use the opcodes listed in Table 1–84 to customize subscription services.

Header File

Various Subscription Management FM opcodes are defined in Header files. 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.

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

Opcode Index

Table 1–84 Subscription Management FM Policy Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS	Adds sharing groups to ordered balance groups. By default, this policy opcode creates or modifies ordered balance groups for profile sharing group members (accounts or services) when the profile sharing group is created or modified. See the discussion on working with profile sharing groups in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE	Creates 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 policy opcode creates an ordered balance group for a service that is a member of a profile sharing group. See the discussion on working with profile sharing groups in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING	Clears fields in a **/service** object at cancellation time, if you customize the opcode. See the discussion on customizing provisioning when canceling a product in *BRM Managing Customers*.	Limited
PCM_OP_SUBSCRIPTION_POL_CONFIG_EET	Defines, for each event category, the event fields that the Event Extraction Manager passes to the event extract output file. See the discussion on customizing how to extract events for rerating in *BRM Setting Up Pricing and Rating*.	Recommended

Table 1–84 (Cont.) Subscription Management FM Policy Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_POL_COUNT_LINES	Allows you to customize how subscriptions are counted when exclusion rules apply for discounts based on the number of subscriptions. See the discussion on discounts based on number of subscriptions in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST	Handles rerating events for BRM's default automatic rerating scenarios. Allows you to customize automatic rerating or handle your own automatic rerating scenarios.	Recommended
PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS	Retrieves data for provisioning from the **provisioning_tags** array of a service-specific configuration object and from the **/config/provisioning_tag** object. See the discussion on getting a list of provisioning tags in *BRM Managing Customers*.	Limited
PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS	Returns a list of all **/sponsorship** objects. See the discussion on getting a list of charges available for charge sharing in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_POL_NOTIFY_AGGREGATION	Allows you to customize how the aggregation counters are updated for discounts based on monthly fees and usage. See the discussion on discounts based on monthly fees and usage in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUBSCRIPTION_POL_PRE_FOLD	Allows customization before folds are applied. See the discussion on applying folds in *BRM Setting Up Pricing and Rating*.	Limited
PCM_OP_SUBSCRIPTION_POL_PREP_FOLD	Prepares the list of resources to be folded after a product cancellation. See the discussion on customizing which resources to fold when products are canceled in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS	Validates the members of a monitor group or profile sharing group. See the discussions on validating the members of a balance monitor group in *BRM Managing Accounts Receivable* and validating profile sharing group members in *BRM Managing Customers*.	Recommended

Table 1–84 (Cont.) Subscription Management FM Policy Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL	Allows customized validation based on account data during deal-to-deal transition. See the discussion on customizing deal transitions in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN	Allows customized validation based on account data during plan-to-plan transition. See the discussion on customizing deal transitions in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING	Sets fields in a **/service** object when a product is purchased, if the opcode is customized. See the discussion on customizing provisioning when a product is purchased in *BRM Managing Customers*.	Limited
PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT	Allows you to customize the distribution of discounts for snowball discounting. See the discussion on customizing snowball discounts in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL	Allows you to specify if a product is canceled or deleted. See the discussion on customizing product cancellation in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT	Allows you to specify if a discount is canceled or deleted. See the discussion on customizing discount cancellation in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL	Allows you to customize the time interval for applying cycle forward or arrears fees. See the discussion on customizing the cycle interval for products in *BRM Setting Up Pricing and Rating*.	Recommended

Table 1–84 (Cont.) Subscription Management FM Policy Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD	Allows you to specify the order in which to fold resources in a balance group. See the discussion on customizing the order to apply folds in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE	Allows you to define custom searches for selecting specific events from the accounts selected for rerating. See the discussion on customizing event searches for selective rerating in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_SUBSCRIPTION_POL_UPDATE_CDC	Allows you to customize how days are counted for discounts based on the number of contract days. See the discussion on discounts based on the number of contract days in *BRM Configuring Pipeline Rating and Discounting*.	Recommended

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS

Adds sharing groups to ordered balance groups. By default, this policy opcode creates or modifies ordered balance groups for profile sharing group members (accounts or services) when the profile sharing group is created or modified.

This policy opcode 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 this policy opcode to use it with other types of sharing groups or to use it with only certain profile sharing groups.

Note: This policy 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, this policy opcode should be listed after the account synchronization publish opcode, PCM_OP_PUBLISH_GEN_PAYLOAD. See the discussion on merging event notification lists in *BRM Developer's Guide*.

This opcode is not called by any opcode.

See the discussion on working with profile sharing groups in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE

Creates 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 policy 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.

This policy opcode is triggered when a new service is added to an existing account.

You can customize this policy opcode to use it with other types of sharing groups or to use it with only certain profile sharing groups.

Note: This policy opcode is listed in the **pin_notify** file for **/event/notification/service/create**.

When merging event notification files, this policy opcode should be listed after the PCM_OP_TCF_PROV_CREATE_SVC_ORDER opcode. See the discussion on merging event notification lists in *BRM Developer's Guide*.

This opcode is not called by any opcode.

See the discussion on working with profile sharing groups in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING

Important: Do not call this policy opcode directly.

Clears fields in a **/service** object at cancellation time, if you customize the opcode.

This opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT standard opcode when a product is canceled.

See the discussion on customizing provisioning when canceling a product in *BRM Managing Customers*.

Note: The BRM provisioning tag framework is the preferred method of customizing provisioning when a product is canceled. See the discussion on using the provisioning tag framework in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_CONFIG_EET

Defines, for each event category, the event fields that the Event Extraction Manager passes to the event extract output file. The default implementation defines the event field to output file mapping for GSM and GPRS event categories only.

If balance monitoring is enabled, this opcode adds the monitor impacts from the event to the output of the Event Extraction Manager.

This opcode is not called by any opcode.

See the discussion on customizing how to extract events for rerating in *BRM Setting Up Pricing and Rating*.

Example 1–282 Sample Input Flist

```
# number of field entries allocated 2, used 2
0 PIN_FLD_POID      POID [0] 0.0.0.1 /event/delayed/session/gsm 8419 0
0 PIN_FLD_CATEGORY   STR [0] "GSM"
```

Example 1–283 Sample Output Flist

```
# number of field entries allocated 2, used 2
0 PIN_FLD_POID      POID [0] 0.0.0.1 /event/delayed/session/gsm 8419 0
0 PIN_FLD_BUFFER     BUF [0] ...
```

PCM_OP_SUBSCRIPTION_POL_COUNT_LINES

Allows you to customize how subscriptions (lines) are counted when exclusion rules apply for discounts based on the number of subscriptions.

Note: By default, this policy opcode is provided as a binary (library) file so you can write, build, and install your own policy implementations. To use this opcode, you must replace it rather than edit an existing source code file.

You can use this opcode to change the way subscriptions are counted, such as changing the count value to a value other than 1. For example, if the count value is set to 2, whenever a subscription is added, the line counter resource balance is incremented by 2 instead of 1.

This opcode is called by the PCM_OP_SUBSCRIPTION_COUNT_LINES standard opcode.

See the discussion on discounts based on number of subscriptions in *BRM Configuring Pipeline Rating and Discounting*.

PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST

Handles rerating events for BRM's default automatic rerating scenarios. Allows you to customize automatic rerating or handle your own automatic rerating scenarios.

This opcode 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.

This opcode is called by event notification.

This policy opcode 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 policy opcode to achieve those results.

This policy opcode also provides a hook for you to analyze any event-notification events you have configured (for trigger-dependent rerating) and determine if a rerate job needs to be created for those events.

This opcode is not called by any opcode.

PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS

Important: Do not call this policy opcode directly.

Retrieves data for provisioning from the **provisioning_tags** array of a service-specific configuration object and from the **/config/provisioning_tag** object.

This opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT standard opcode when a product is canceled.

For more information on retrieving provisioning information, see the discussion on getting a list of provisioning tags in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS

Returns a list of all **/sponsorship** objects.

This opcode is not called by any opcode.

See the discussion on getting a list of charges available for charge sharing in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_POL_NOTIFY_AGGREGATION

Allows you to customize how the aggregation balance is updated for discounts based on monthly fees and usage.

> **Note:** By default, this opcode is provided as a binary (library) file so you can write, build, and install your own policy implementations. To use this opcode, you must replace it rather than edit an existing source code file.

This opcode is called by the PCM_OP_SUBSCRIPTION_NOTIFY_AGGREGATION standard opcode.

See the discussion on discounts based on monthly fees and usage in *BRM Configuring Pipeline Rating and Discounting*.

PCM_OP_SUBSCRIPTION_POL_PRE_FOLD

Allows customization before folds are applied.

By default, this policy opcode is an empty hook provided to facilitate any customization prior to the folding currency or noncurrency resources. For example, when billing is run, this policy opcode is called to verify that the **pin_cycle_fees** have been charged to an account.

This opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT and PCM_OP_BILL_MAKE_BILL standard opcodes.

See the discussion on applying folds in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_PREP_FOLD

Prepares the list of resources that have to be folded when a product is cancelled.

This opcode is called by the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS standard opcodes.

See the discussion on customizing which resources to fold when products are canceled in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS

Validates the members of a monitor group or profile sharing group. For profile sharing groups, this policy opcode has no default validation rules, but rules can be implemented by customizing it.

This policy opcode takes as input the list of potential members and returns only those members that pass validation.

This opcode is called by the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE and PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY standard opcodes.

See the following discussions:

- Validating the members of a balance monitor group in *BRM Managing Accounts Receivable*
- Validating profile sharing group members in *BRM Managing Customers*

Example 1–284 Sample Input Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 59967 10
0 PIN_FLD_GROUP_OBJ          POID [0] 0.0.0.1 /group/sharing/monitor 121
0 PIN_FLD_PARENT             POID [0] 0.0.0.1 /account 59967 10
0 PIN_FLD_MEMBERS           ARRAY [0] allocated 2, used 2
1    PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 48832 0
1    PIN_FLD_SERVICE_OBJ     POID [0] 0.0.0.1 /service/ip/gprs 3974 0
```

Example 1–285 Sample Output Flist

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 59967 10
0 PIN_FLD_GROUP_OBJ          POID [0] 0.0.0.1 /group/sharing/monitor 121
0 PIN_FLD_PARENT             POID [0] 0.0.0.1 /account 59967 10
0 PIN_FLD_MEMBERS           ARRAY [0] allocated 2, used 2
1    PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 48832 0
1    PIN_FLD_SERVICE_OBJ     POID [0] 0.0.0.1 /service/ip/gprs 3974 0
```

PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL

Allows customized validation based on account data during deal-to-deal transition. For example, you may restrict a deal transition to customers from a particular location, or require that customers own the first deal for specific period before allowing the transition to a different deal.

This opcode is called by the PCM_OP_SUBSCRIPTION_TRANSITION_DEAL standard opcode.

See the discussion on customizing deal transitions in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN

Allows customized validation based on account data during plan-to-plan transition. For example, you can restrict a plan transition to customers from a particular location, or require that customers own the first plan for specific period before allowing the transition to a different plan.

This opcode is called by the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN standard opcode.

See the discussion on customizing deal transitions in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING

Sets fields in a **/service** object when a product is purchased, if the opcode is customized.

Use PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING to customize product provisioning when a product is purchased. This opcode can be customized to set fields in a **/service** object.

This opcode is called by the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT standard opcode when a product is purchased.

See the discussion on customizing provisioning when a product is purchased in *BRM Managing Customers*.

Note: The BRM provisioning tag framework is the preferred method of customizing provisioning when a product is purchased. See the discussion on using the provisioning tag framework in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT

Allows you to specify the distribution of group discounts to group members. You can modify this policy opcode code to specify an algorithm for distributing the total group discount grant to the individual group members. For instance, you can specify distribution of the group discount based on group member contribution.

This opcode is not called by any opcode.

See the discussion on customizing snowball discounts in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL

Allows you to customize the actions taken for a product cancellation.

This opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT standard opcode to determine the action to take for a product cancellation. Possible actions are:

- To cancel the product and delete the **/purchased_product** object.
- To cancel the product but do not delete the **/purchased_product** object.
- To stop the product cancellation.

See the discussion on customizing product cancellation in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT

Allows you to specify whether a discount is canceled or deleted. You can customize this policy opcode to do one of the following actions:

- Set the status of the **/purchased_discount** object to *canceled* but not delete it.
- Delete the **/purchased_discount** object.
- Stop the discount cancellation.

This opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT standard opcode.

See the discussion on customizing discount cancellation in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL

Allows you to customize the time interval for applying cycle forward and cycle arrears fees for a specified product.

By default, this policy opcode is an empty hook that facilitates customization of the cycle forward and cycle arrears start and end dates for a specific product.

This opcode is called by the PCM_OP_SUBSCRIPTION_APPLY_RATE, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD, and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS standard opcodes.

See the discussion on customizing the cycle interval for products in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD

Allows you to specify the order in which to fold resources in a balance group. For example, you can fold resources in descending order of the resource IDs. By default, resources are folded in ascending order based on the resource ID.

This opcode is called by the PCM_OP_SUBSCRIPTION_CYCLE_FOLD standard opcode.

See the discussion on customizing the order to apply folds in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE

Allows you to define custom searches for rerating events for selected accounts.

This policy opcode is called when the **pin_rerate** utility is used with **-r** parameter to indicate selective rerating.

This opcode is called by the PCM_OP_SUBCRIPTION_RERATE_REBILL standard opcode.

See the discussion on customizing event searches for selective rerating in *BRM Configuring and Running Billing*.

Example 1–286 Sample Input Flist

The following sample input flist shows selective rerating based on event type specified by using the **-n** option with the **pin_rerate** utility, on cycle forward monthly events.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_FLAGS           INT [0] 256
0 PIN_FLD_ARGS          ARRAY [1] allocated 20, used 1
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 12983 0
0 PIN_FLD_ARGS          ARRAY [2] allocated 20, used 1
1     PIN_FLD_END_T        TSTAMP [0] (1041408000) Wed Jan 01 00:00:00 2003
0 PIN_FLD_ARGS          ARRAY [3] allocated 20, used 1
1     PIN_FLD_CREATED_T    TSTAMP [0] (0) <null>
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 10
1     PIN_FLD_POID           POID [0] NULL poid pointer
1     PIN_FLD_CREATED_T    TSTAMP [0] (0) <null>
1     PIN_FLD_EFFECTIVE_T  TSTAMP [0] (0) <null>
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_SERVICE_OBJ    POID [0] NULL poid pointer
1     PIN_FLD_ACCOUNT_OBJ    POID [0] NULL poid pointer
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] NULL pin_decimal_t ptr
1     PIN_FLD_RERATE_OBJ     POID [0] NULL poid pointer
1     PIN_FLD_BAL_IMPACTS   ARRAY [*]     NULL array ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [*]     NULL array ptr
0 PIN_FLD_ARGS          ARRAY [4] allocated 20, used 1
1     PIN_FLD_
POID           POID [0] 0.0.0.1 /event/billing/product/fee/cycle/cycle_forward
        monthly -1 0
0 PIN_FLD TEMPLATE        STR [0] "select X from /event where F1 =
    V1 and F2 >=V2 and ( F4 = V4 ) orde  by F2 asc, F3 asc "
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 12983 0
```

PCM_OP_SUBSCRIPTION_POL_UPDATE_CDC

Allows you to customize the criteria for updating the contract days counter for discounts based on the number of contract days. For example, you can have updates take effect in the next billing cycle.

> **Note:** By default, this opcode is provided as a binary (library) file so you can write, build, and install your own policy implementations. To use this opcode, you must replace it rather than edit an existing source code file.

This opcode is called by the PCM_OP_SUBSCRIPTION_UPDATE_CDC standard opcode.

See the discussion on discounts based on the number of contract days in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–287 Sample Input Flist

The following example input flists show the different kinds of input that can be sent to this opcode, depending on the type of event.

This example shows the input from a change in the status of a subscription service.

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /event/customer/status -1 0
0 PIN_FLD_NAME               STR [0] "Customer Mngmt. Event Log"
0 PIN_FLD_USERID            POID [0] 0.0.0.1 /service/admin_client 2 179
0 PIN_FLD_SESSION_OBJ       POID [0] 0.0.0.1 /event/session 220553236438860771 0
0 PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account 20451 0
0 PIN_FLD_PROGRAM_NAME       STR [0] "Automatic Account Creation"
0 PIN_FLD_START_T         TSTAMP [0] (1084073173) Sun May 9 08:56:13 2004
0 PIN_FLD_END_T           TSTAMP [0] (1084073173) Sun May 9 08:56:13 2004
0 PIN_FLD_SYS_DESCR          STR [0] "Set Status (acct)"
0 PIN_FLD_STATUSES         ARRAY [0] allocated 20, used 3
1      PIN_FLD_STATUS               ENUM [0] 0
1      PIN_FLD_STATUS_FLAGS          INT [0] 0
1      PIN_FLD_CLOSE_WHEN_T       TSTAMP [0] (1083209265) Thu Apr 29 08:57:45 2004
0 PIN_FLD_STATUSES         ARRAY [1] allocated 20, used 3
1      PIN_FLD_STATUS               ENUM [0] 10100
1      PIN_FLD_STATUS_FLAGS          INT [0] 0
1      PIN_FLD_CLOSE_WHEN_T       TSTAMP [0] (0) <null>
0 PIN_FLD_EARNED_START_T   TSTAMP [0] (0) <null>
0 PIN_FLD_EARNED_END_T     TSTAMP [0] (0) <null>
0 PIN_FLD_EARNED_TYPE         INT [0] 0
0 PIN_FLD_EFFECTIVE_T       TSTAMP [0] (0) <null>
```

The following input flist is generated when billing starts for the **/billinfo** object that is associated with the subscription service.

```
0 PIN_FLD_POID              POID [0] 0.0.0.0 /event/notification/billing/start -1 0
0 PIN_FLD_NAME               STR [0] "bracket event created"
0 PIN_FLD_USERID            POID [0] 0.0.0.1 /service/pcm_client 1 1
0 PIN_FLD_SESSION_OBJ       POID [0] 0.0.0.1 /event/session 220553236438860715 0
0 PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account 16251 0
0 PIN_FLD_PROGRAM_NAME       STR [0] "pin_bill_accts"
0 PIN_FLD_START_T         TSTAMP [0] (1101148200) Tue Nov 23 00:00:00 2004
0 PIN_FLD_END_T           TSTAMP [0] (1103740200) Thu Dec 23 00:00:00 2004
0 PIN_FLD_SYS_DESCR          STR [0] "bracket event created"
```

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 12177 0
0 PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0 PIN_FLD_SERVICES        ARRAY [0] allocated 20, used 4
1      PIN_FLD_POID                    POID [0] 0.0.0.1 /service/email 8785 -1
1      PIN_FLD_LOGIN                    STR [0] "ac1"
1      PIN_FLD_PASSWD_CLEAR             STR [0] "password"
1      PIN_FLD_INHERITED_INFO      SUBSTRUCT [0] allocated 20, used 1
2           PIN_FLD_SERVICE_EMAIL         SUBSTRUCT [0] allocated 20, used 1
3                PIN_FLD_PATH                     STR [0] "/tmp"
```

Example 1–288 Sample Output Flist

```
0 PIN_FLD_POID              POID [0] 0.0.0.0 /event/notification/billing/start -10
0 PIN_FLD_RESULTS          ARRAY [0] allocated 20, used 4
1      PIN_FLD_POID             POID [0] 0.0.0.1 /event/billing/debit 8785 -1
(the debit event created as a result of update to CDC balance)
```

Subscription Management FM Standard Opcodes

The opcodes in Table 1–85 are used to manage subscription services.

Header File

Various Subscription Management FM opcodes are defined in Header files. 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.

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

Opcode Index

Table 1–85 Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_CALC_BEST_PRICING	Calculates the best price for an account for a billing cycle. See the discussion on calculating the best price by using the best pricing opcode in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_SUBSCRIPTION_CANCEL_DEAL	Cancels all products and discounts owned by a deal. See the discussion on canceling deals in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT	Cancels a discount bundled in a deal. See the discussion on canceling discounts in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT	Cancels a product owned by an account or service storable class. See the discussion on canceling products in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION	Cancels a subscription service. See the discussion on canceling a subscription service in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_CHANGE_DEAL	Changes the subscription products associated with an account. See the discussion on how deals are modified in BRM Managing Customers.	Recommended
PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS	Validates product option changes. See the discussion on validating changes to deals in BRM Managing Customers.	Recommended

Table 1–85 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_COUNT_LINES	Counts the number of active subscriptions when exclusion rules apply for discounts based on number of subscriptions. See the discussion on discounts based on number of subscriptions in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUBSCRIPTION_CYCLE_ARREARS	Applies cycle arrears charges to an account. See the discussion on applying cycle arrears fees in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_SUBSCRIPTION_CYCLE_FOLD	Applies cycle fold events for an account. See the discussion on applying folds in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_SUBSCRIPTION_CYCLE_FORWARD	Applies cycle forward charges or refunds to an account. See the discussion on applying cycle forward fees in *BRM Setting Up Pricing and Rating*.	Last Resort
PCM_OP_SUBSCRIPTION_GET_HISTORY	Retrieves event history for an account's deals, products, and services. See the discussion on finding events associated with deals, products, discounts, and services in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS	Reads the purchased products and discounts filtered under the scope of an account, billinfo or a service passed in the input. See the discussion on reading data for all valid purchased products and discounts in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP	Creates, modifies, or deletes the ordered balance group for an account or service that is part of a resource, profile, or monitor sharing group. See the discussions on managing ordered balance groups in *BRM Managing Accounts Receivable*, adding a profile group to a member's ordered balance group in *BRM Managing Customers*, or adding a monitor group to a member's ordered_balgrp object in *BRM Managing Accounts Receivable*.	Recommended

Table 1–85 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY	Creates one or more ordered balance groups for an account or service and modifies the priority of the resource sharing groups included in each ordered balance group. See the discussion on managing ordered balance groups in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE	Creates the rate change object. See the discussion on tracking rate changes for rerating in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_SUBSCRIPTION_PROVISION_ERA	Creates, modifies, or deletes **/profile** objects as part of a provisioning tag. See the discussion on configuring provisioning tags in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_SUBSCRIPTION_PURCHASE_DEAL	Purchases the products and discounts in a deal for the account or service object specified in the input flist. See the discussion on how deals are purchased in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT	Allows purchase of discount instances bundled in a deal. See the discussion on purchasing discounts in *BRM Managing Customers*.	Limited
PCM_OP_SUBSCRIPTION_PURCHASE_FEES	Applies deferred purchase fees to a product. See the discussion on applying deferred product purchase fees in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT	Purchases a product for an account or service. See the discussion on how products are purchased in *BRM Managing Customers*.	Limited
PCM_OP_SUBSCRIPTION_RATE_CHANGE	Creates rerating requests when there is a rate change in a cycle. See the discussion on rerating cycle fees in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS	Retrieves hierarchical relationships of deals, products, discounts, and services for each account. See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_RERATE_REBILL	For each account, rerates the events, which have been identified by **pin_rerate,** from a specified start date. See the discussion on how comprehensive rerating works in *BRM Configuring and Running Billing*.	Recommended

Table 1–85 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER	Transfers services from one balance group to another. See the discussion on transferring services between balance groups by using custom client applications in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SET_BUNDLE	Creates, modifies, and deletes **/purchased_bundle** objects. See the discussion on adding Siebel CRM promotion names to invoices in *BRM Configuring and Running Billing*.	Recommended
PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO	Modifies the cycle information about a discount in a deal. See the discussion on modifying discount attributes in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS	Changes the status of a discount in a deal. See the discussion on how BRM changes discount status in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_SET_PRODINFO	Customizes existing product information for a specific account. See the discussion on how products are modified in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS	Sets product status and status flags. See the discussion on how BRM changes product status in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE	Creates a discount sharing group, charge sharing group, profile sharing group, or monitor sharing group. See the discussion on creating resource sharing groups in *BRM Managing Accounts Receivable*, creating/ profile sharing/ groups in *BRM Managing Customers*, or creating, modifying, or deleting/ group/ sharing/ monitor objects in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE	Deletes a discount sharing group, charge sharing group, profile sharing group, or monitor sharing group. See the discussion on deleting resource sharing groups in *BRM Managing Accounts Receivable*, deleting profile sharing groups in *BRM Managing Accounts Receivable*, or creating, modifying, or deleting /group/sharing/monitor objects in *BRM Managing Accounts Receivable*.	Recommended

Table 1–85 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY	Modifies a discount sharing group, charge sharing group, profile sharing group, or monitor sharing group. See the discussion son modifying resource sharing groups in *BRM Managing Accounts Receivable*, modifying profile sharing groups in *BRM Managing Customers*, or creating, modifying, or deleting /group/sharing/monitor objects in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT	Changes the owner of a discount sharing group, charge sharing group, profile sharing group, or monitor sharing group. See the discussion on changing the owner of a resource sharing group in *BRM Managing Accounts Receivable*, changing the owner of a profile sharing group through a customized client application in *BRM Managing Customers*, or changing the owner of a balance monitor in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER	Adds a member to a sponsor group. See the discussion on adding a member to a sponsor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_CREATE	Creates a sponsored group. See the discussion on creating a sponsor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE	Deletes a sponsored group. See the discussion on deleting a sponsor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER	Deletes a member from a sponsored group. See the discussion on deleting a member from a sponsor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_MODIFY	Modifies information in the **/group/sponsor** storable class. See the discussion on modifying a sponsor group in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT	Assigns a new sponsor group owner to a sponsor group. See the discussion on setting the parent of a sponsor group in *BRM Managing Accounts Receivable*.	Recommended

Table 1–85 (Cont.) Subscription Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER	Transfers rollover resources to another account or service. See the discussion on transferring rollover resources in *BRM Managing Accounts Receivable*.	Recommended
PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION	Transfers a subscription service to another subscriber account. See the discussion on transferring a subscription service in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_TRANSITION_DEAL	Transitions one deal to another. See the discussion on how deals are transitioned in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_TRANSITION_PLAN	Transitions one plan to another. See the discussion on transitioning plans in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY	Validates deal-to-deal transitions. See the discussion on validating deal transitions in *BRM Managing Customers*.	Recommended
PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY	Validates a discount to see if it can be used with other discounts or with plans. See the discussion on validating discount dependencies in *BRM Managing Customers*.	Recommended

PCM_OP_SUBSCRIPTION_CALC_BEST_PRICING

Calculates the best price by comparing the base deal with alternate deals in a best pricing configuration.

See the discussion on calculating the best price by using the best pricing opcode in *BRM Configuring and Running Billing*.

Return Values

This opcode follows the standard mechanism of setting error buffer on failure.

PCM_OP_SUBSCRIPTION_CANCEL_DEAL

Cancels ownership of a deal for a specified account or service.

This opcode is called when a deal is canceled. This opcode cancels all products and discounts associated with the specific deal and then cancels the deal itself.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated deal cancellation when certain conditions are met.

If the deal purchase is successful, it returns the **/account** POID and the POID of the **/event/billing/deal/cancel** event.

See the following discussions:

- Canceling deals in *BRM Managing Customers*
- Backdated deal, product, and discount in *BRM Configuring and Running Billing*

PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT

Cancels the discount instances associated with the **/account** object or **/service** object specified in the input flist.

> **Note:** If the **/service** object specified is NULL, all the discount instances associated with the **/account** object are canceled, if not, only the discount instances associated with the **/service** object are canceled.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated discount cancellation when certain conditions are met.

See the following discussions:

- Canceling discounts in *BRM Managing Customers*
- Backdated deal, product, and discount cancellation in *BRM Configuring and Running Billing*

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT

Cancels the products for the **/account** object specified in the input flist.

This opcode is recursively called by PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel each product associated with a specific deal.

See the discussion on canceling products in *BRM Managing Customers*.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated product cancellation when certain conditions are met.

PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION

Cancels a subscription service and its member services.

See the discussion on canceling a subscription service in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL

Changes the products associated with a deal for an account.

See the discussion on how deals are modified in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS

Validates changes to deals; for example, to check for prerequisite or mutually exclusive deals. This opcode first attempts to add a service to an account. If successful, it then adds or removes deals as needed.

See the discussion on validating changes to deals in *BRM Managing Customers*.

Example 1–289 Sample Input Flist for Adding a Service

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 42992 0
0 PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0 PIN_FLD_PLAN_OBJ         POID [0] 0.0.0.1 /plan 15692 0
0 PIN_FLD_SERVICES        ARRAY [0] allocated 20, used 5
1     PIN_FLD_DEALS       ARRAY [0] allocated 20, used 2
2         PIN_FLD_DEAL_OBJ POID [0] 0.0.0.1 /deal 12876 0
2         PIN_FLD_BOOLEAN   INT [0] 0
1     PIN_FLD_SERVICE_ID    STR [0] "Test1"
1     PIN_FLD_SERVICE_OBJ  POID [0] 0.0.0.1 /service/email -1 0
1     PIN_FLD_LOGIN         STR [0] "AB-a8-1"
1     PIN_FLD_PASSWD_CLEAR   STR [0] "AB-a8-1"
```

Example 1–290 Sample Output Flist for Adding a Service

```
0 PIN_FLD_POID                POID [0] 0.0.0.1 /account 42992 0
0 PIN_FLD_RESULTS            ARRAY [0] allocated 6, used 6
1     PIN_FLD_SERVICES       ARRAY [0] allocated 5, used 5
2         PIN_FLD_DEALS      ARRAY [0] allocated 2, used 2
3             PIN_FLD_DEAL_OBJ POID [0] 0.0.0.1 /deal 12876 0
3             PIN_FLD_BOOLEAN   INT [0] 0
2         PIN_FLD_SERVICE_ID    STR [0] "Test1"
2         PIN_FLD_SERVICE_OBJ  POID [0] 0.0.0.1 /service/email 43193 0
2         PIN_FLD_LOGIN         STR [0] "ab-a8-1@corp.portal.com"
2         PIN_FLD_PASSWD_CLEAR  STR [0] "AB-a8-1"
1     PIN_FLD_POID             POID [0] 0.0.0.1 /plan 15692 0
1     PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account 42992 0
1     PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
1     PIN_FLD_START_T        TSTAMP [0] (1062813523) Fri Sep 05 18:58:43 2003
1     PIN_FLD_END_T           TSTAMP [0] (1062813523) Fri Sep 05 18:58:43 2003
```

Example 1–291 Sample Input Flist for Adding a Deal

```
0 PIN_FLD_POID                                POID [0] 0.0.0.1 /account 4241767 0
0 PIN_FLD_PLAN_OBJ                            POID [0] 0.0.0.1 /plan 3215264 0
0 PIN_FLD_PROGRAM_NAME                         STR [0] "Customer Center"
0 PIN_FLD_SERVICES                            ARRAY [0] allocated 8, used 8
1   PIN_FLD_SERVICE_OBJ                        POID [0] 0.0.0.1 /service/ip 4240615
0
1   PIN_FLD_BAL_GRP_OBJ                        POID [0] 0.0.0.1 /balance_group
4243303 0
1   PIN_FLD_BOOLEAN                             INT [0] 1
1   PIN_FLD_SERVICE_ID                          STR [0] "/service/ip33020888"
1   PIN_FLD_DEALS                              ARRAY [0] allocated 2, used 2
2      PIN_FLD_BOOLEAN                          INT [0] 0
2      PIN_FLD_DEAL_INFO                    SUBSTRUCT [0] allocated 7, used 7
3         PIN_FLD_PRODUCTS                      ARRAY [0] allocated 20, used 20
4            PIN_FLD_PURCHASE_END_T           TSTAMP [0] (0) <null>
4            PIN_FLD_PURCHASE_START_T         TSTAMP [0] (0) <null>
4            PIN_FLD_QUANTITY                DECIMAL [0] 3.00
4            PIN_FLD_USAGE_END_UNIT              INT [0] 0
```

```
4            PIN_FLD_USAGE_END_OFFSET          INT [0] 0
4            PIN_FLD_USAGE_START_UNIT          INT [0] 0
4            PIN_FLD_USAGE_START_OFFSET        INT [0] 0
4            PIN_FLD_CYCLE_END_UNIT            INT [0] 0
4            PIN_FLD_CYCLE_END_OFFSET          INT [0] 0
4            PIN_FLD_CYCLE_START_UNIT          INT [0] 0
4            PIN_FLD_CYCLE_START_OFFSET        INT [0] 0
4            PIN_FLD_PURCHASE_END_UNIT         INT [0] 0
4            PIN_FLD_PURCHASE_END_OFFSET       INT [0] 0
4            PIN_FLD_PURCHASE_START_UNIT       INT [0] 0
4            PIN_FLD_PURCHASE_START_OFFSET     INT [0] 0
4            PIN_FLD_PLAN_OBJ                 POID [0] 0.0.0.1 /plan 3215264 0
4            PIN_FLD_PRODUCT_OBJ              POID [0] 0.0.0.1 /product 14476 0
4            PIN_FLD_USAGE_DISCOUNT        DECIMAL [0] 0
4            PIN_FLD_CYCLE_DISCOUNT        DECIMAL [0] 0
4            PIN_FLD_PURCHASE_DISCOUNT     DECIMAL [0] 0
4            PIN_FLD_STATUS                   ENUM [0] 1
4            PIN_FLD_STATUS_FLAGS              INT [0] 0
4            PIN_FLD_USAGE_END_T            TSTAMP [0] (0) <null>
4            PIN_FLD_USAGE_START_T          TSTAMP [0] (0) <null>
4            PIN_FLD_CYCLE_END_T            TSTAMP [0] (0) <null>
4            PIN_FLD_CYCLE_START_T          TSTAMP [0] (0) <null>
3       PIN_FLD_NAME                           STR [0] "Deal 2a - Unlimited
Internet Service"
3       PIN_FLD_POID                          POID [0] 0.0.0.1 /deal 12364 0
3       PIN_FLD_END_T                        TSTAMP [0] (0) <null>
3       PIN_FLD_FLAGS                           INT [0] 0
3       PIN_FLD_START_T                      TSTAMP [0] (0) <null>
3       PIN_FLD_DESCR                           STR [0] ""
3             PIN_FLD_PLAN_OBJ           POID [0] 0.0.0.1 /plan 3215264 0
1     PIN_FLD_SUBSCRIPTION_INDEX    INT [0] 0
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0  0 0
1     PIN_FLD_BAL_INFO_INDEX    INT [0] 0
```

Example 1–292 Sample Output Flist for Adding a Deal

```
0 PIN_FLD_POID       POID [0] 0.0.0.1 /account 4241767 0
0 PIN_FLD_RESULTS   ARRAY [0] allocated 4, used 4
1   PIN_FLD_POID     POID [0] 0.0.0.1 /account 4241767 0
1   PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
2      PIN_FLD_POID   POID [0] 0.0.0.1 /event/billing/product/fee/cycle/cycle_
forward_monthly 216383888346359225 0
1   PIN_FLD_RESULTS ARRAY [1] allocated 1, used 1
2      PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/action/purchase
17592186088889 0
1   PIN_FLD_RESULTS ARRAY [2] allocated 1, used 1
2      PIN_FLD_POID   POID [0] 0.0.0.1 /event/billing/deal/purchase 17592186086329
0
```

Example 1–293 Sample Input Flist for Customizing Products

```
0 PIN_FLD_POID             POID [0] 0.0.0.1 /account 40965 0
0 PIN_FLD_PROGRAM_NAME      STR [0] "testnap"
0 PIN_FLD_PLAN_OBJ         POID [0] 0.0.0.1 /plan 8494 0
0 PIN_FLD_SERVICES        ARRAY [0] allocated 20, used 3
1     PIN_FLD_SERVICE_OBJ  POID [0] 0.0.0.1 /service/ip 43525 5
1     PIN_FLD_SERVICE_ID    STR [0] "Test1"
1     PIN_FLD_DEALS       ARRAY [0] allocated 20, used 2
2         PIN_FLD_BOOLEAN         INT [0] 0
2         PIN_FLD_DEAL_INFO SUBSTRUCT [0] allocated 20, used 7
3             PIN_FLD_POID           POID [0] 0.0.0.1 /deal 11822 0
```

```
3             PIN_FLD_NAME             STR [0] "PTP CFM PP PC 30$"
3             PIN_FLD_DESCR            STR [0] ""
3             PIN_FLD_START_T       TSTAMP [0] (0) <null>
3             PIN_FLD_END_T         TSTAMP [0] (0) <null>
3             PIN_FLD_FLAGS            INT [0] 0
3             PIN_FLD_PRODUCTS        RRAY [0] allocated 41, used 24
4                 PIN_FLD_PRODUCT_OBJ            POID [0] 0.0.0.1 /product 8750 0
4                 PIN_FLD_QUANTITY            DECIMAL [0] 1
4                 PIN_FLD_DESCR                   STR [0] ""
4                 PIN_FLD_PURCHASE_START_T      TSTAMP [0] (86393) Thu Jan 01
15:59:53 1970
4                 PIN_FLD_PURCHASE_END_T        TSTAMP [0] (0) <null>
4                 PIN_FLD_PURCHASE_DISCOUNT    DECIMAL [0] 0
4                 PIN_FLD_PURCHASE_DISC_AMT    DECIMAL [0] 0
4                 PIN_FLD_PURCHASE_FEE_AMT     DECIMAL [0] 0
4                 PIN_FLD_CYCLE_START_T         TSTAMP [0] (86393) Thu Jan 01
15:59:53 1970
4                 PIN_FLD_CYCLE_END_T           TSTAMP [0] (0) <null>
4                 PIN_FLD_CYCLE_DISCOUNT       DECIMAL [0] 0
4                 PIN_FLD_CYCLE_DISC_AMT       DECIMAL [0] 0
4                 PIN_FLD_CYCLE_FEE_AMT        DECIMAL [0] 0
4                 PIN_FLD_USAGE_START_T         TSTAMP [0] (86393) Thu Jan 01
15:59:53 1970
4                 PIN_FLD_USAGE_END_T           TSTAMP [0] (0) <null>
4                 PIN_FLD_USAGE_DISCOUNT       DECIMAL [0] 0
4                 PIN_FLD_STATUS                  ENUM [0] 1
4                 PIN_FLD_STATUS_FLAGS             INT [0] 0
4                 PIN_FLD_PURCHASE_START_CYCLE DECIMAL [0] 0
4                 PIN_FLD_PURCHASE_END_CYCLE   DECIMAL [0] 0
4                 PIN_FLD_CYCLE_START_CYCLE    DECIMAL [0] 0
4                 PIN_FLD_CYCLE_END_CYCLE      DECIMAL [0] 0
4                 PIN_FLD_USAGE_START_CYCLE    DECIMAL [0] 0
4                 PIN_FLD_USAGE_END_CYCLE       DECIMAL [0] 0
```

Example 1–294 Sample Output Flist for Customizing Products

```
0 PIN_FLD_POID         POID [0] 0.0.0.1 /account 40965 0
0 PIN_FLD_RESULTS     ARRAY [0] allocated 3, used 3
1     PIN_FLD_POID     POID [0] 0.0.0.1 /account 40965 0
1     PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/action/purchase
17592186088069 0
1     PIN_FLD_RESULTS ARRAY [1] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/deal/purchase
17592186087045 0
```

PCM_OP_SUBSCRIPTION_COUNT_LINES

Updates the count of active subscriptions for discounts based on the number of subscription services.

This opcode is used when applying discounts based on a number of subscriptions that consider discount exclusion rules.

See the discussion on discounts based on number of subscriptions in *BRM Configuring Pipeline Rating and Discounting*.

Error Handling

This opcode reports standard BRM error conditions.

Return Value

If successful, this opcode returns the contents of the output flist, including the results information from the event element and the POID of the event object.

If this opcode fails, it returns an error in the error buffer.

PCM_OP_SUBSCRIPTION_CYCLE_ARREARS

Applies cycle arrears fees to an account.

Note: Cycle arrears fees are applied only for a single month.

It returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_arrears** event.

See the discussion on applying cycle arrears fees in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_CYCLE_FOLD

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 the discussion on applying folds in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_CYCLE_FORWARD

Applies cycle forward charges or refunds to an account. For example, this opcode is called when a product or discount is purchased, canceled, activated, or inactivated.

If successful, it returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_forward_type** event.

When balance monitoring is enabled, this opcode passes the PIN_FLD_MONITORS array to the PCM_OP_ACT_USAGE opcode.

See the following discussions:

- Applying cycle forward fees in *BRM Setting Up Pricing and Rating*
- Balance monitoring in *BRM Managing Accounts Receivable*

PCM_OP_SUBSCRIPTION_GET_HISTORY

Retrieves the event history for a deal, product, discount, or service instance associated with an account.

See the discussion on finding events associated with deals, products, discounts, and services in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS

Retrieves the purchased products and discounts associated with an account. Because products and discounts are part of the account object, reading the account object fetches all the purchased products and discounts.

This opcode requires a scope object in the PIN_FLD_SCOPE_OBJ field of the input flist. The scope object can be an **/account, /billinfo,** or **/service** object:

- **/account object:** Fetches all the products and discounts for the account and its services.
- **/billinfo object:** Fetches products and discounts assoicated with the specified bill unit.
- **/service object:** Fetches products and discounts that belong to the specified service.

This opcode performs a search based on the values in the input flist. It repeats the search if more result sets must be fetched. For example, if a service object is passed as the scope object, the input flist looks like this:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 618010 0
0 PIN_FLD_SCOPE_OBJ       POID [0] 0.0.0.1 /service/ip 615706 3
0 PIN_FLD_STATUS_FLAGS    INT [0] 3
0 PIN_FLD_END_T           TSTAMP [0] (1154415600) Tue Aug  1 00:00:00 2006
0 PIN_FLD_VALIDITY_FLAGS  INT [0] 3
0 PIN_FLD_INCLUSION_FLAGS INT [0] 2
0 PIN_FLD_OVERRIDE_FLAGS  INT [0] 2
0 PIN_FLD_OVERRIDDEN_OBJ  POID [0] 0.0.0.1 /purchased_product 324706 0
0 PIN_FLD_DEAL_OBJ        POID [0] 0.0.0.1 /deal 615702 3
0 PIN_FLD_PACKAGE_ID      INT [0] 23
0 PIN_FLD_PRODUCTS        ARRAY [*]     NULL array ptr
0 PIN_FLD_DISCOUNTS       ARRAY [*]     NULL array ptr
```

The input for the opcode also contains qualifiers to fetch the correct set of offerings:

- To specify whether to fetch only products or only discounts, use the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays.
- To fetch a limited number of fields, specify those fields under the PRODUCTS and DISCOUNTS arrays.
- To fetch products and discounts valid as of a specified time, pass the PIN_FLD_END_T field 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: The opcode fetches only active offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_INACTIVE: The opcode fetches only inactive offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_CLOSED: The opcode fetches only closed offerings.

Note: Using multiple values implies the target object can satisfy any of them.

- PIN_FLD_VALIDITY_FLAGS:
  - PIN_SUBS_FLG_OFFERING_VALIDITY_CYCLE: The opcode compares the specified END_T value with the CYCLE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_PURCHASE: The opcode compares the specified END_T value with the PURCHASE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_USAGE: The opcode compares the specified END_T value with the USAGE_END_T value.

  Note: Using multiple flags implies the target object must satisfy all of them.

- PIN_FLD_INCLUSION_FLAGS:
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_PRODS: Includes all eligible products (both account and subscription products).
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_DISCS: Includes all eligible discounts (both account and subscription discounts).

  Note: When this field is omitted, only eligible offerings from the specified scope are returned.

- PIN_FLD_OVERRIDE_FLAGS:
  - PIN_SUBS_FLG_OFFERING_ACCT_LEVEL_ONLY: Filters out only account offerings. Valid for **/account** objects only.
  - PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY: When a valid offering POID is sent, returns all the offerings that override the input offering. When a NULL offering POID is sent, only the base products are returned. This flag must be used with the OVERRIDDEN _OBJ field. It is valid for any scope.

  Note: When none of these flags is present or when this field is omitted, all products are returned.

- PIN_FLD_OVERRIDDEN_OBJ: Use this when handling tailor-made products, which override their base products and can be searched for if the PIN_FLD_OVERRIDE_FLAGS field is set to PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY. In that case, set this parameter to the base product to search for all the products that use that base product, including the base product itself. If OVERRIDDEN_OBJ is null and the OVERRIDE_FLAGS value is set to PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY, this opcode fetches only base products.
- PIN_FLD_PACKAGE_ID: Limits a search by the package_ID, which translates to a single plan. This can be used with any scope.
- PIN_FLD_DEAL_OBJ: Limits your search to objects that are part of the deal specified by the deal object POID.

See the discussion on reading data for all valid purchased products and discounts in *BRM Managing Customers*.

Example 1–295 Sample Input Flist

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /account 216663 10
0 PIN_FLD_SCOPE_OBJ POID [0] 0.0.0.1 /service/ip 214231 0
0 PIN_FLD_STATUS_FLAGS   INT [0] 3
0 PIN_FLD_VALIDITY_FLAGS INT [0] 3
0 PIN_FLD_PRODUCTS ARRAY [0]
1     PIN_FLD_OFFERING_OBJ POID [0] 0.0.0.1 /purchased_product 123456 0
1     PIN_FLD_PLAN_OBJ       POID [0] NULL
1     PIN_FLD_PRODUCT_OBJ    POID [0] NULL
1     PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_DISC_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) 0
1     PIN_FLD_PURCHASE_FEE_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_START_T TSTAMP [0] 0
0 PIN_FLD_DISCOUNTS ARRAY [0]
1     PIN_FLD_OFFERING_OBJ POID [0] NULL
1     PIN_FLD_PLAN_OBJ       POID [0] NULL
1     PIN_FLD_DISCOUNT_OBJ    POID [0] NULL
1     PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_DISC_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) 0
1     PIN_FLD_PURCHASE_FEE_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_START_T TSTAMP [0] 0
```

Example 1–296 Sample Output Flist

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /account 216663 10
0 PIN_FLD_PRODUCTS  ARRAY [0] allocated 31, used 31
1     PIN_FLD_OFFERING_OBJ POID [0] 0.0.0.1 /purchased_product 215235
1     PIN_FLD_PLAN_OBJ  POID [0] 0.0.0.1 /plan 215975 0
1     PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 215463
1     PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_DISC_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
1     PIN_FLD_PURCHASE_FEE_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_START_T TSTAMP [0] (1130745600)
0 PIN_FLD_PRODUCTS      ARRAY [1] allocated 31, used 31
1     PIN_FLD_OFFERING_OBJ POID [0] 0.0.0.1 /purchased_product 21531 0
1     PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.1 /plan 215975 0
1     PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 21512 0
1     PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_DISC_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
1     PIN_FLD_PURCHASE_FEE_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_START_T TSTAMP [0] (1123234500)
0 PIN_FLD_DISCOUNTS      ARRAY [0] allocated 31, used 31
1     PIN_FLD_OFFERING_OBJ POID [0] 0.0.0.1 /purchased_discount 2118 5
1     PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.1 /plan 215975 0
1     PIN_FLD_DISCOUNT_OBJ    POID [0] 0.0.0.1 /discount 5463
1     PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_DISC_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
1     PIN_FLD_PURCHASE_FEE_AMT DECIMAL [0] 0
1     PIN_FLD_PURCHASE_START_T TSTAMP [0] (1130745600)
```

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP

Creates, modifies, or deletes the ordered balance group (**/ordered_balgrp** object) for an account or service that is a member of a resource, profile, or monitor 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 and charge sharing groups, the rank controls the order in which the group's resource balances are impacted by events.

For profile sharing and monitor sharing groups, the rank is not significant.

See the following discussions:

- Adding a profile group to a member's ordered balance group in *BRM Managing Customers*
- Adding a monitor group to a member's **/ordered_balgrp** object in *BRM Managing Accounts Receivable*
- Managing ordered balance groups in *BRM Managing Accounts Receivable*

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 resource, profile, or monitor sharing groups included in one or more existing ordered balance groups.

For discount sharing and charge sharing groups, the rank controls the order in which the group's resource balances are impacted by events.

For profile sharing and monitor sharing groups, the rank is not significant.

See the following discussions:

- Adding a profile group to a member's ordered balance group in *BRM Managing Customers*
- Adding a monitor group to a member's **/ordered_balgrp** object in *BRM Managing Accounts Receivable*
- Managing ordered balance groups in *BRM Managing Accounts Receivable*

PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE

Creates the **/rate_change** object, which stores details about the products affected by a rate change, including the rate tiers and rate plans for the product.

When you change a cycle fee by adding a new rate tier in Pricing Center, the event notification feature triggers this opcode. This opcode reads the products 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.

See the discussion on tracking rate changes for rerating in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_PROVISION_ERA

This opcode creates, modifies, or deletes **/profile** objects. When specified in a **/config/provisioning_tag** object, this opcode runs when a product or discount containing the provisioning tag is purchased or canceled. Profiles can store extended rating attributes (ERAs) and other custom attributes.

This opcode calls PCM_OP_CUST_CREATE_PROFILE, PCM_OP_CUST_MODIFY_PROFILE, or PCM_OP_CUST_DELETE_PROFILE, depending on the action it needs to take.

When creating a profile, this opcode creates a **/profile/acct_extrating** object for an account-level profile and a **/profile/serv_extrating** object for a service-level profile.

See the discussion on configuring provisioning tags in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL

Purchases the products and discounts in a deal 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 deal *POID* because no actual BRM deal exists.

If the deal was created in BRM, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls the PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY opcode to validate deal-to-deal dependency rules. If products and discounts were created in an external CRM, this validation does not take place.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated deal purchases when certain conditions are met.

See the following discussions:

- How deals are purchased in *BRM Managing Customers*
- Backdated deal, product, and discount purchase in *BRM Configuring and Running Billing*

PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT

Allows purchase of a discount bundled in a deal.

See the discussion on purchasing discounts in *BRM Managing Customers*.

Important: Do not call this opcode directly. This opcode is always called by the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated discount purchases when certain conditions are met.

See the discussion on backdated deal, product, and discount purchase in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_PURCHASE_FEES

Applies deferred purchase fees for a product with an expired purchase start time.

By default, purchase fees are applied at the time of product purchase. However, you can defer the purchase fees to a later date. For example, a subscriber can sign up for a product that is not available until a later date. The product's purchase fees are deferred and applied when the product becomes available.

See the discussion on applying deferred product purchase fees in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT

Purchases a product for an account or service.

See the discussion on how products are purchased in *BRM Managing Customers*.

Important: Do not call this opcode directly. This opcode is always called by the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated product purchases when certain conditions are met.

See the discussion on backdated deal, product, and discount purchase in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_RATE_CHANGE

Creates rerating requests when there is a cycle forward or cycle forward arrears event rate change in the middle of the current cycle. This opcode uses event notification to trigger rerating.

> **Note:** Rerating is not triggered for cycle_arrears rate changes or rate changes in future cycles.

When you run the **pin_rate_change** utility after a rate change, the utility calls this opcode and provides details about the accounts and products affected by the rate change.

This opcode 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 (resulting in the rerate job objects used by the **pin_rerate** utility).

See the discussion on rerating cycle fees in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS

Retrieves the hierarchical relationships of deals, products, discounts, and services for an account.

For example, this opcode is used by Customer Center when a request is made to view a list of an account's deals, products, discounts, and services in a hierarchical format.

See the discussion on getting plans, deals, and products for purchase in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_RERATE_REBILL

Rerates events for a specified account.

This opcode 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.

See the discussion on how comprehensive rerating works in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER

A wrapper opcode that performs all the tasks necessary to transfer a service from one balance group to another.

This opcode is called by Customer Center or a custom client application.

See the discussion on transferring services between balance groups by using custom client applications in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SET_BUNDLE

Manages **/purchased_bundle** objects in the BRM database. Use this opcode to add promotion names and details to BRM invoices.

See the discussion on adding Siebel CRM promotion names to invoices in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO

Modifies or sets a discount's purchase, cycle, or usage date information.

This opcode is called, for example, when a discount is set to inactive status when purchased and is activated later.

See the discussion on modifying discount attributes in *BRM Managing Customers*.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated discount status changes when certain conditions are met.

See the discussion on backdated deal, product, and discount purchase in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS

Changes the status of a **/purchased_discount** object in a deal for an account or service.

This opcode is called when the status of a discount is changed. This can occur:

- When the status of the account or service that owns the discount is changed. In this case, the discount's status is changed to the status of the account or service.
- When the status of a discount is changed from inactive to active.

See the discussion on how BRM changes discount status in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_SET_PRODINFO

Changes the information for a specified product in an account.

If the PCM_OPFLG_CALC_ONLY flag is set, this opcode 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.

See the discussion on modifying products in *BRM Managing Customers*.

When automatic rerating is enabled, this opcode triggers automatic rerating of backdated product status changes when certain conditions are met.

See the discussion on backdated deal, product, and discount purchase in *BRM Configuring and Running Billing*.

PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS

Sets the product status of a **/purchased_product** object owned by an account.

This opcode is called:

- When the status of an account or service is changed.
- When a product status is changed. You might need to change only the product status itself; for example, the product was purchased as inactive because of future provisioning, and you activate it later.

See the discussion on how BRM changes product status in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE

Creates the following types of sharing groups for an account or service:

- Discount or charge sharing group: See the discussion on creating resource sharing groups in *BRM Managing Accounts Receivable*.
- Profile sharing group: See the discussion on creating profile sharing groups in *BRM Managing Customers*.
- Monitor sharing group: See the discussion on creating, modifying, or deleting **/group/sharing/monitor** objects in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE

Deletes the following:

- For a discount sharing group, the opcode deletes shared discounts, group members, or the sharing group itself.
- For a charge sharing group, the opcode deletes sponsored charges, group members, or the sharing group itself.
- For a profile sharing group, the opcode deletes shared profiles, group members, or the sharing group itself.
- For monitor sharing groups, the opcode deletes the group itself.

If successful, this opcode returns the POID of the sharing group object that was modified and the POID of the event that was generated.

See the discussion on deleting resource sharing groups in *BRM Managing Accounts Receivable*, deleting profile sharing groups in *BRM Managing Customers*, or creating, modifying, or deleting /group/sharing/monitor objects in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY

Modifies sharing groups as follows:

- Adds shared discounts, sponsored charges, or group members to a resource sharing group.
- Adds shared profiles or group members to a profile sharing group.
- Modifies monitor sharing groups.

If successful, this opcode returns the POID of the group that was modified and the POIDs of the events that were generated to record the group modification.

See the discussion on modifying resource sharing groups in *BRM Managing Accounts Receivable*, codifying profile sharing groups in *BRM Managing Customers*, or creating, modifying, or deleting /group/sharing/monitor objects in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT

Changes the owner of a charge sharing, discount sharing, profile sharing, or monitor sharing group.

If successful, this opcode returns the POID of the sharing group that was modified and the event that was generated to record the parent change.

See the discussion on changing the owner of a resource sharing group in *BRM Managing Accounts Receivable*, changing the owner of a profile sharing group through a customized client application in *BRM Managing Customers*, or changing the owner of a balance monitor in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER

Adds a member to a sponsored group.

If successful, this opcode returns these values:

- The POID of the **/group/sponsor** object to which the member was added.
- The POID of the **/event/group/member** object created to record adding the member to the sponsored group.

See the discussion on adding a member to a sponsor group in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_CREATE

Creates a sponsored group.

If successful, this opcode returns these values:

- The POID of the **/group/sponsor** object created.
- The POID of the **/event/group/parent** object created to record the creation of the sponsored group.

See the discussion on creating a sponsor group in *BRM Managing Accounts Receivable*.

Example 1–297 Sample Input Flist

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /account 11107 14
0 PIN_FLD_NAME        STR [0] "E.T. Telecom"
```

Example 1–298 Sample Output Flist

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /group/sponsor 10451 0
0 PIN_FLD_RESULTS        ARRAY [0] allocated 20, used 1
1 PIN_FLD_POID        POID [0] 0.0.0.1 /event/group/parent 9427 0
```

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE

Deletes a sponsored group.

If successful, this opcode returns the POID of the **/group/sponsor** object that was passed in the input flist.

See the discussion on deleting a sponsor group in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER

Deletes a member from a sponsored group.

If successful, this opcode returns these values:

- The POID of the **/group/sponsor** object passed in the input flist.
- The POID of the **/event/group/member** object created to record the deletion of the sponsored group member.

See the discussion on deleting a member from a sponsor group in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_MODIFY

Modifies the product and rate information sponsored by the sponsor group.

If successful, this opcode returns the POID of the **/group/sponsor** object passed in the input flist.

See the discussion on modifying a sponsor group in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT

Assigns a new sponsor group owner to a sponsor group.

If successful, this opcode returns the POID of the event **/group/member/parent** that is created.

See the discussion on setting the parent of a sponsor group in *BRM Managing Accounts Receivable*.

Example 1–299 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /group/sponsor 10423 0 <-- Sponsor Group
POID
0 PIN_FLD_PROGRAM_NAME    STR [0] "Sample"
0 PIN_FLD_PARENT         POID [0] 0.0.0.1 /account 9559 0 <-- Account POID of the
new (intended) sponsor group owner
```

Example 1–300 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 9559 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 1, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/group/parent
204438794022169271 0
```

PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER

Checks the rollover-transfer profile object, **/profile/rollover_transfer**, to make sure it is configured and valid for the resource and receiver and then transfers the entire rollover amount to the receiver.

This opcode is called by the event notification system when the **/event/billing/cycle/rollover/monthly** and **/event/billing/cycle/rollover_correction** events occur.

See the discussion on transferring rollover resources in *BRM Managing Accounts Receivable*.

PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION

Transfers a subscription service to another subscriber account.

Use this opcode to transfer a subscription service and its member services from one subscriber to another subscriber's account.

See the discussion on transferring a subscription service in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_TRANSITION_DEAL

Transitions a deal from one account to another.

See the discussion on how deals are transitioned in *BRM Managing Customers*.

Example 1–301 Sample Input Flist

```
00 PIN_FLD_POID           POID [0] 0.0.0.1 /account 41349 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0 PIN_FLD_TRANSITION_TYPE   ENUM [0] 1
0 PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip 43813 0
0 PIN_FLD_FROM_DEAL_INFO SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PACKAGE_ID         INT [0] 12345
0 PIN_FLD_TO_DEAL_INFO SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 42980 0
```

Example 1–302 Sample Output Flist

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /account 41349 0
0 PIN_FLD_RESULTS    ARRAY [0] allocated 5, used 5
1     PIN_FLD_POID    POID [0] 0.0.0.1 /account 41349 0
1     PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/fee/cycle/cycle_
forward_monthly 216383888346360741 0
1     PIN_FLD_RESULTS ARRAY [1] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/fee/cancel
216383888346359717 0
1     PIN_FLD_RESULTS ARRAY [2] allocated 1, used 1
2         PIN_FLD_POID POID [0] 0.0.0.1 /event/billing/product/action/cancel
17592186085477 0
1     PIN_FLD_RESULTS ARRAY [3] allocated 1, used 1
2         PIN_FLD_POID POID [0] 0.0.0.1 /event/billing/deal/cancel 17592186087525
0
0 PIN_FLD_RESULTS     ARRAY [1] allocated 5, used 5
1     PIN_FLD_POID     POID [0] 0.0.0.1 /account 41349 0
1     PIN_FLD_RESULTS ARRAY [0] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/fee/purchase
216383888346358885 0
1     PIN_FLD_RESULTS ARRAY [1] allocated 1, used 1
2         PIN_FLD_POID  POID [0] 0.0.0.1 /event/billing/product/fee/cycle/cycle_
forward_monthly 216383888346360933 0
1     PIN_FLD_RESULTS  ARRAY [2] allocated 1, used 1
2         PIN_FLD_POID   POID [0] 0.0.0.1 /event/billing/product/action/purchase
17592186085989 0
1     PIN_FLD_RESULTS  ARRAY [3] allocated 1, used 1
2         PIN_FLD_POID   POID [0] 0.0.0.1 /event/billing/deal/purchase
17592186088037 0
```

PCM_OP_SUBSCRIPTION_TRANSITION_PLAN

Transitions one plan to another.

This opcode takes as input a source plan that specifies the plan currently owned by the account and a target plan that specifies the plan to transition to.

See the discussion on transitioning plans in *BRM Managing Customers*.

Example 1–303 Sample Input Flist

This example shows an account upgrade from /plan 13842 to /plan 15890. The /deal 15954, /deal 14866, and /service/email are added to the account.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 15186 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
0 PIN_FLD_FROM_PLAN      POID [0] 0.0.0.1 /plan 13842 0
0 PIN_FLD_TO_PLAN        POID [0] 0.0.0.1 /plan 15890 0
0 PIN_FLD_TRANSITION_TYPE   ENUM [0] 1
0 PIN_FLD_SERVICES      ARRAY [0] allocated 20, used 2
1     PIN_FLD_FROM_SERVICE SUBSTRUCT [0] allocated 20, used 1
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip 12754 0
1     PIN_FLD_TO_SERVICE   SUBSTRUCT [0] allocated 20, used 5
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip -1 0
2         PIN_FLD_SERVICE_ID      STR [0] "ip4"
2         PIN_FLD_LOGIN           STR [0] "ip4_a"
2         PIN_FLD_PASSWD_CLEAR    STR [0] "ip4_a"
2         PIN_FLD_DEALS         ARRAY [0] allocated 20, used 1
3             PIN_FLD_PACKAGE_ID      INT [0] 12345
3             PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 14866 0
0 PIN_FLD_SERVICES      ARRAY [1] allocated 20, used 1
1     PIN_FLD_TO_SERVICE   SUBSTRUCT [0] allocated 20, used 5
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/email -1 0
2         PIN_FLD_SERVICE_ID      STR [0] "ip4_1"
2         PIN_FLD_LOGIN           STR [0] "ip4_a"
2         PIN_FLD_PASSWD_CLEAR    STR [0] "ip4_a"
2         PIN_FLD_DEALS         ARRAY [0] allocated 20, used 1
3              PIN_FLD_PACKAGE_ID         INT [0] 12345
3             PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 15954 0
```

Example 1–304 Sample Output Flist

This example shows the return flists from PCM_OP_SUBSCRIPTION_CANCEL_DEAL, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL, PCM_OP_CUST_SET_STATUS, and PCM_OP_CUST_MODIFY_CUSTOMER called to cancel and purchase deals to perform the plan upgrade.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 15186 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 4, used 4
1    PIN_FLD_POID           POID [0] 0.0.0.1 /account 15186 0
1    PIN_FLD_RESULTS       ARRAY [0] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1
/event/billing/product/fee/cycle/cycle_forward_monthly 14514 0
1    PIN_FLD_RESULTS       ARRAY [1] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1
/event/billing/product/action/cancel 15538 0
1    PIN_FLD_RESULTS       ARRAY [2] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/deal/cancel 12978
0
0 PIN_FLD_RESULTS       ARRAY [1] allocated 4, used 4
1    PIN_FLD_POID           POID [0] 0.0.0.1 /account 15186 0
```

```
1    PIN_FLD_RESULTS       ARRAY [0] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1
/event/billing/product/fee/cycle/cycle forward_monthly 15026 0
1    PIN_FLD_RESULTS       ARRAY [1] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1
/event/billing/product/action/purchase 14002 0
1    PIN_FLD_RESULTS       ARRAY [2] allocated 1, used 1
2        PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/deal/purchase
16050 0
0 PIN_FLD_RESULTS       ARRAY [2] allocated 7, used 7
1    PIN_FLD_SERVICES      ARRAY [1] allocated 6, used 6
2        PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/email 12722 0
2        PIN_FLD_SERVICE_ID      STR [0] "ip4_1"
2        PIN_FLD_LOGIN           STR [0] "ip4_a@corp.portal.com"
2        PIN_FLD_PASSWD_CLEAR    STR [0] "ip4_a"
2        PIN_FLD_DEALS         ARRAY [0] allocated 1, used 1
3             PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 14866 0
2        PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 12498 0
1    PIN_FLD_POID           POID [0] 0.0.0.1 /plan 15890 0
1    PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 15186 0
1    PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
1    PIN_FLD_END_T        TSTAMP [0] (1117609200) Wed Jun 01 00:00:00 2005
1    PIN_FLD_START_T      TSTAMP [0] (1117609200) Wed Jun 01 00:00:00 2005
1    PIN_FLD_ACCTINFO      ARRAY [0] allocated 3, used 3
2        PIN_FLD_POID           POID [0] 0.0.0.1 /account 15186 17
2        PIN_FLD_CURRENCY        INT [0] 840
2        PIN_FLD_CURRENCY_SECONDARY    INT [0] 0
```

PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY

Validates deal-to-deal dependency rules.

This opcode is called by Customer Center and PCM_OP_CUST_SET_STATUS.

See the discussion on validating deal transitions in *BRM Managing Customers*.

PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY

Validates the discount with other discounts or plans. Mutually exclusive dependencies are configured in the **/dependency** storable class.

See the discussion on validating discount dependencies in *BRM Managing Customers*.

Suspense Manager FM Standard Opcodes

The opcodes listed in Table 1–86 manage suspended EDRs stored in the BRM database as **/suspended_usage** objects.

For information about suspense manager, see the discussion on Suspense Manager in *BRM Configuring Pipeline Rating and Discounting*.

Header File

Include the **ops/suspense.h** header file in all applications that call these opcodes. For details, see the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 1–86 Suspense Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUSPENSE_DEFERRED_DELETE	Deletes records for suspended EDRs after Revenue Assurance has been completed. Available with Suspense Manager. See the discussion on deleting suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_DELETE_USAGE	Deletes records for suspended EDRs. Available with Suspense Manager. See the discussion on deleting records for suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_EDIT_USAGE	Changes the contents of fields in suspended EDRs. Available with Suspense Manager. See the discussion on changing the contents of fields in suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_RECYCLE_USAGE	Initiates EDR recycling. See the discussion on initiating suspense recycling in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_SEARCH_DELETE	Deletes call records with a specific recycle key and a status of **succeeded** or **written off**. See the discussion on deleting call records with a specific recycle key and a status of Succeeded or Written-off in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_SEARCH_EDIT	Changes fields in a large number of suspended records in one database operation. See the discussion on editing suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.	Recommended

Table 1–86 (Cont.) Suspense Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUSPENSE_SEARCH_RECYCLE	Recycles suspended EDRs. Available with Suspense Manager. See the discussion on recycling suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_SEARCH_WRITE_OFF	Writes off a large number of suspended records in one database operation. See the discussion on writing off suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_UNDO_EDIT_USAGE	Undoes edits to suspended EDRs. Available with Suspense Manager. See the discussion on changing the contents of fields in suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.	Recommended
PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE	Writes off suspended EDRs. See the discussion on writing off suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.	Recommended

PCM_OP_SUSPENSE_DEFERRED_DELETE

Deletes EDRs in a written off state or succeeded state. This opcode is scheduled to execute at a later time to ensure Revenue Assurance.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on deleting suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.

PCM_OP_SUSPENSE_DELETE_USAGE

Deletes EDRs in a written off state or succeeded state.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on deleting records for suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–305 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /admin_action/suspended_
usage/1rec 0 0
0 PIN_FLD_PROGRAM_NAME                  STR [0] "TestNap"
0 PIN_FLD_SUSPENDED_USAGE_OBJS          ARRAY [0] allocated 13, used 13
1   SUSPENDED_USAGE_OBJ                 POID [0] 0.0.0.1 /suspended_usage/telco
15204 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS          ARRAY [1] allocated 13, used 13
1   SUSPENDED_USAGE_OBJ                 POID [0] 0.0.0.1 /suspended_usage/telco
15588 0
```

PCM_OP_SUSPENSE_EDIT_USAGE

Changes the contents of EDR fields for a suspended call record. The Suspense Management Center calls this opcode to edit a suspended call record.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on changing the contents of fields in suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–306 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit -1 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "TestNap"
0 PIN_FLD_EDITS         ARRAY [0] allocated 4, used 4
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 1, used 1
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 1, used 1
3             PIN_FLD_CALLED_TO       STR [0] "111"
1     PIN_FLD_OLD_VALUE       STR [0] ""
1     PIN_FLD_NEW_VALUE       STR [0] "111"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.B_NUMBER"
0 PIN_FLD_EDITS         ARRAY [1] allocated 4, used 4
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 1, used 1
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 1, used 1
3             PIN_FLD_CALLING_FROM    STR [0] "111"
1     PIN_FLD_OLD_VALUE       STR [0] ""
1     PIN_FLD_NEW_VALUE       STR [0] "111"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.A_NUMBER"
0 PIN_FLD_EDITS         ARRAY [2] allocated 4, used 4
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 1, used 1
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 1, used 1
3             PIN_FLD_CALL_DURATION DECIMAL [0] 111.00
1     PIN_FLD_OLD_VALUE       STR [0] ""
1     PIN_FLD_NEW_VALUE       STR [0] "111"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.DURATION"
0 PIN_FLD_EDITS         ARRAY [3] allocated 4, used 4
1     PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 1, used 1
2         PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 1, used 1
3             PIN_FLD_START_TIME   TSTAMP [0] (1079006993) Thu Mar 11 12:09:53
2004
1     PIN_FLD_OLD_VALUE       STR [0] ""
1     PIN_FLD_NEW_VALUE       STR [0] "20040311173953"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.CHARGING_START_TIMESTAMP"
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [0] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897081 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [1] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897082 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [2] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897083 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [3] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897084 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [4] allocated 1, used 1
```

```
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897085 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [5] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897086 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS  ARRAY [6] allocated 1, used 1
1     PIN_FLD_SUSPENDED_USAGE_OBJ   POID [0] 0.0.0.1 /suspended_usage/telco
1308577166727897087 0
```

Example 1–307 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit -1 0
0 PIN_FLD_POIDS         ARRAY [0] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit
58432 0
0 PIN_FLD_POIDS         ARRAY [1] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit
60480 0
0 PIN_FLD_POIDS         ARRAY [2] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit
57920 0
0 PIN_FLD_POIDS         ARRAY [3] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit
59968 0
0 PIN_FLD_RESULT         ENUM [0] 0
```

PCM_OP_SUSPENSE_RECYCLE_USAGE

Initiates EDR recycling. During recycling, suspended EDRs are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to recycle suspended EDRs.

See the discussion on initiating suspense recycling in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–308 Sample Input Flist

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /admin_action/suspended_
usage/1rec 0 0
0 PIN_FLD_PROGRAM_NAME                  STR [0] "TestNap"
0 PIN_FLD_RECYCLE_MODE                  ENUM [0] 1
0 PIN_FLD_SUSPENDED_USAGE_OBJS          ARRAY [0] allocated 13, used 13
1   PIN_FLD_SUSPENDED_USAGE_OBJ         POID [0] 0.0.0.1 /suspended_usage/telco
12530 0
0 PIN_FLD_SUSPENDED_USAGE_OBJS          ARRAY [1] allocated 13, used 13
1   PIN_FLD_SUSPENDED_USAGE_OBJ         POID [0] 0.0.0.1 /suspended_usage/telco
13298 0
```

PCM_OP_SUSPENSE_SEARCH_DELETE

Deletes call records with a status of **succeeded** or **written off** that match criteria specified in the input flist. 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.

See the discussion on deleting call records with a specific recycle key and a status of Succeeded or Written-off in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–309 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_PROGRAM_NAME   STR [0]  "test client"
0 PIN_FLD_ARGS          ARRAY [1]
1     PIN_FLD_POID        POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_ARGS          ARRAY [2]
1     PIN_FLD_POID        POID [0] 0.0.0.1 /suspended_usage/telco/% -1 0
0 PIN_FLD_ARGS          ARRAY [3]
1     PIN_FLD_FILENAME        STR [0] "test_MED1.edr"
0 PIN_FLD_ARGS          ARRAY [4]
1     PIN_FLD_POID        STR [0] 0.0.0.1 /suspended_usage <poid range start>
0 PIN_FLD_ARGS          ARRAY [5]
1     PIN_FLD_POID        STR [0] 0.0.0.1 /suspended_usage <poid range end>
0 PIN_FLD_TEMPLATE        STR [0] "( F1 = V1 or F2 like V2 )  and F3 = V3 and F4
>= V4 and F5 <= V5"
```

Example 1–310 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_COUNT       INT [0] 1000
```

To search for and recycle suspended call records containing a specific recycle key, use PCM_OP_SUSPENSE_EDIT_USAGE.

See the discussion on changing the contents of fields in suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

PCM_OP_SUSPENSE_SEARCH_EDIT

This opcode makes changes to a large number of suspended records that meet the criteria specified in the input template.

See the discussion on editing suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–311 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_PROGRAM_NAME   STR [0]  "test client"
0 PIN_FLD_FLAGS            INT [0]  512
0 PIN_FLD_ARGS          ARRAY [1]
1     PIN_FLD_POID         POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_ARGS          ARRAY [2]
1     PIN_FLD_POID         POID [0] 0.0.0.1 /suspended_usage/telco/% -1 0
0 PIN_FLD_ARGS          ARRAY [3]
1     PIN_FLD_FILENAME        STR [0] "test_MED1.edr"
0 PIN_FLD_ARGS          ARRAY [4]
1     PIN_FLD_POID         STR [0] 0.0.0.1 /suspended_usage <poid range start>
0 PIN_FLD_ARGS          ARRAY [5]
1     PIN_FLD_POID         STR [0] 0.0.0.1 /suspended_usage <poid range end>
0 PIN_FLD_TEMPLATE         STR [0] "( F1 = V1 or F2 like V2 )  and F3 = V3 and F4
>= V4 and F5 <= V5"
0 PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
2         PIN_FLD_CALLED_TO       STR [0] "004941067601"
2         PIN_FLD_CALLING_FROM    STR [0] "00491732411"
0 PIN_FLD_EDITS         ARRAY [0] allocated 20, used 4
1     PIN_FLD_NEW_VALUE       STR [0] "004941067601"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.B_NUMBER"
0 PIN_FLD_EDITS         ARRAY [1] allocated 20, used 4
1     PIN_FLD_NEW_VALUE       STR [0] "00491732411"
1     PIN_FLD_FIELD_NAME      STR [0] "DETAIL.A_NUMBER"
```

Example 1–312 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_POIDS        ARRAY [0]
1     PIN_FLD_POID    POID [0] 0.0.0.1 /admin_action/suspended_usage/edit 111184 0
0 PIN_FLD_POIDS        ARRAY [1]
1     PIN_FLD_POID    POID [0] 0.0.0.1 /admin_action/suspended_usage/edit 111185 0
0 PIN_FLD_COUNT       INT [0] 1000
```

PCM_OP_SUSPENSE_SEARCH_RECYCLE

Searches for and queues suspended call records for recycling based on criteria specified in the input flist. You can specify the following criteria:

- A recycle key
- A CDR file
- A search template

See the discussion on recycling suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–313 Sample Input Flist

```
# number of field entries allocated 5, used 4
0 PIN_FLD_POID           POID [0] 0.0.0.1 /dummy -1 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "pin_recycle tool"
0 PIN_FLD_MODE           ENUM [0] 1
0 PIN_FLD_RECYCLE_KEY     STR [0] "tb"
```

Example 1–314 Sample Output Flist

If successful, this output flist returns the POID of the **/admin_action/suspended_usage/recycle** object created for the recycled call records.

```
# number of field entries allocated 20, used 2
0 PIN_FLD_POID           POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_COUNT           INT [0] 0
```

PCM_OP_SUSPENSE_SEARCH_WRITE_OFF

This opcode writes off a large number of suspended records that match the search criteria in the input flist.

See the discussion on writing off suspended records in bulk in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–315 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /suspended_usage/telco -1 0
0 PIN_FLD_PROGRAM_NAME   STR [0]  "test client"
0 PIN_SEARCH_INFO        SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_TEMPLATE    STR [0] "username"
1     PIN_FLD_FLAGS       INT [0] 0
1     PIN_FLD_ARGS        ARRAY[2] "test_MED1.edr"
2         PIN_FLD_RESULTS_LIMIT   INT [0] "004941067601"
2         PIN_FLD_RESULTS         ARRAY [0] allocated 20, used 1
```

Example 1–316 Sample Output Flist

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /admin_action/suspended_usage/writeoff 111183 0
```

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE

Undoes edits to suspended call records used by Suspense Manager. This opcode is called by Suspense Management Center to perform the undo edit action. It replaces the value of a field in a suspended call record with the value in that field before the last edit was made.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on undoing edits to suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–317 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit 60800
0
```

Example 1–318 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /admin_action/suspended_usage/edit 59968
0
0 PIN_FLD_RESULT         ENUM [0] 0
0 PIN_FLD_ACTION_OBJ     POID [0] NULL poid pointer
0 PIN_FLD_COUNT           INT [0] 4
```

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE

Writes off suspended EDRs. When a suspended EDR is written off, they cannot be edited or recycled.

Important: This opcode is available to Suspense Manager customers only.

See the discussion on writing off suspended EDRs in *BRM Configuring Pipeline Rating and Discounting*.

Example 1–319 Sample Input Flist

```
0 PIN_FLD_POID                      POID [0] 0.0.0.1 /admin_action/suspended_
usage/1rec 0 0
0 PIN_FLD_PROGRAM_NAME              STR [0] "TestNap"
0 PIN_FLD_SUSPENDED_OBJS            ARRAY [0] allocated 13, used 13
1   PIN_FLD_SUSPENDED_OBJ           POID [0] 0.0.0.1 /suspended_usage/telco
15204 0
0 PIN_FLD_SUSPENDED_OBJS            ARRAY [1] allocated 13, used 13
1   PIN_FLD_SUSPENDED_OBJ           POID [0] 0.0.0.1 /suspended_usage/telco
15588 0
```

System Manager FM Standard Opcodes

The opcodes in Table 1–87 are used to manage the administration of BRM servers.

Header File

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

Opcode Index

Table 1–87 System Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_INFMGR_ADD_OBJECT	Adds objects to the System Manager. This opcode is not supported at this time.	Not supported
PCM_OP_INFMGR_CANCEL_DOWNTIME	Cancels scheduled downtime for server.	Recommended
PCM_OP_INFMGR_DELETE_OBJECT	Deletes objects from the System Manager. This opcode is not supported at this time.	Not supported
PCM_OP_INFMGR_GET_INFO	Gets information about servers.	Recommended
PCM_OP_INFMGR_GET_LOGLEVEL	Dynamically gets the log level of the Connection Manager (CM) at run time.	Recommended
PCM_OP_INFMGR_GET_STATUS	Gets status of servers.	Recommended
PCM_OP_INFMGR_MODIFY_MONITOR_INTERVAL	Modifies the monitoring interval for status of BRM servers.	Limited
PCM_OP_INFMGR_SATELLITE_CM_START_FORWARDING	Tells a satellite CM to start or resume passing opcodes to the main CM.	Recommended
PCM_OP_INFMGR_SATELLITE_CM_STOP_FORWARDING	Tells a satellite CM to stop passing opcodes to the main CM.	Recommended
PCM_OP_INFMGR_SCHEDULE_DOWNTIME	Schedules downtime for server.	Recommended
PCM_OP_INFMGR_SET_LOGLEVEL	Dynamically sets the log level of the CM and DM to the value specified in the input flist.	Recommended
PCM_OP_INFMGR_START_SERVER	Starts servers.	Recommended
PCM_OP_INFMGR_STOP_SERVER	Stops servers.	Recommended

PCM_OP_INFMGR_ADD_OBJECT

Adds a node object to System Manager, which then reads information about servers that are running on that node, and starts to monitor that node.

Note: This opcode is not supported at this time.

PCM_OP_INFMGR_CANCEL_DOWNTIME

Cancels any scheduled downtime for a server.

You can specify the name of a server, as specified in System Manager's configuration file, in the array PIN_FLD_ARGS. At present, this array can have only one PIN_FLD_SERVER_NAME element.

The System Manager **cdt** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_DELETE_OBJECT

Deletes a node object from System Manager, which then stops managing and monitoring the servers running on that node.

Note: This opcode is not supported at this time.

PCM_OP_INFMGR_GET_INFO

Gets information about servers running on a node or in a cell.

The cell, node, or server name is the name specified in the Node Manager's configuration file. This command accepts only one cell, node, or server name.

If the object type is **CELL**, the output includes one PIN_FLD_CELLS for each cell. In each PIN_FLD_CELLS is one or more PIN_FLD_NODES.

If the object type is **NODE** or **SERVER**, the output is one or more PIN_FLD_NODES fields.

The System Manager **gi** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_GET_LOGLEVEL

Dynamically gets the log level of the Connection Manager (CM) at run-time.

Example 1–320 Sample Input Flist

To get the log level of the CM at run-time, send this flist to the opcode:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_FLAGS           INT [0] 1
```

Example 1–321 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_LOGLEVEL        INT [0] 3
```

PCM_OP_INFMGR_GET_STATUS

Gets the status of servers running on a node or in a cell.

The cell, node, or server name is the name specified in the Node Manager's configuration file. This command accepts only one cell, node, or server name.

If the object type is **CELL**, the output includes one PIN_FLD_CELLS for each cell. In each PIN_FLD_CELLS is one or more PIN_FLD_NODES.

If the object type is **NODE** or **SERVER**, the output is one or more PIN_FLD_NODES fields.

The System Manager **gs** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_MODIFY_MONITOR_INTERVAL

Modifies the monitoring interval. By default, the interval is two minutes. Status of the servers is sent to the System Manager every interval.

PCM_OP_INFMGR_SATELLITE_CM_START_FORWARDING

Tells a satellite CM to start or resume passing opcodes to the main CM. System Manager searches its configuration file to look for the specified satellite CM, its host, and its port number. If it cannot find the information, the opcode returns an error.

> **Note:** System Manager does not keep the satellite CM's name, host, or port number in memory. Therefore, you do not have to stop and restart System Manager each time you add a new satellite CM to the configuration file.

The System Manager **sfw** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_SATELLITE_CM_STOP_FORWARDING

Tells a satellite CM to stop passing opcodes to the main CM. System Manager searches its configuration file to look for the specified satellite CM, its host, and its port number. If it cannot find the information, the opcode returns an error.

> **Note:** System Manager does not keep the satellite CM's name, host, or port number in memory. Therefore, you do not have to stop and restart System Manager each time you add a new satellite CM to the configuration file.

The System Manager **fwe** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_SCHEDULE_DOWNTIME

Schedules downtime for a server.

System Manager keeps the downtime information locally. Therefore, the server can still be up and running during its scheduled downtime.

When responding to PCM_OP_INFMGR_GET_STATUS, System Manager returns the server's state of maintenance and its scheduled downtime. If the scheduled downtime expires during the next refresh interval (two minutes), the server's state is updated.

Each server can have only one scheduled downtime. Each time PCM_OP_INFMGR_SCHEDULE_DOWNTIME runs, it overwrites any previously scheduled downtime.

The System Manager **sdt** command calls this opcode.

See the discussion on System Manager command-line interface in *BRM Developer's Guide*.

PCM_OP_INFMGR_SET_LOGLEVEL

Dynamically sets or changes the log level of the CM and the debug flags of the DM.

This opcode takes as input the following data:

- The component name, CM or DM.
- The CM log level you want to set in the PIN_FLD_LOGLEVEL field.
- The DM debug level you want to set in the PIN_FLD_FLAGS field.

The values you set using this opcode apply to all the subsequent opcodes called.

For the CM, the value should be an integer from **0** to **3**. If the integer is outside this range, the log level is not changed and a debug message is logged to **cm.pinlog**.

Example 1–322 Sample Input Flists

To set the DM debug flags:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_COMPONENT       STR [0] "DM"
0 PIN_FLD_DEBUG_FLAG    ARRAY [1]
1   PIN_FLD_NAME          STR [0] "DM_DEBUG"
1   PIN_FLD_FLAGS         INT [0] 255
0 PIN_FLD_DEBUG_FLAG    ARRAY [2]
1   PIN_FLD_NAME          STR [0] "DM_DEBUG2"
1   PIN_FLD_FLAGS         INT [0] 4090
0 PIN_FLD_DEBUG_FLAG    ARRAY [3]
1   PIN_FLD_NAME          STR [0] "DM_DEBUG3"
1   PIN_FLD_FLAGS         INT [0] 65535
0 PIN_FLD_DEBUG_FLAG    ARRAY [4]
1   PIN_FLD_NAME          STR [0] "DM_DEBUG4"
1   PIN_FLD_FLAGS         INT [0] 2
```

Note: The PIN_FLD_FLAGS field can be either INT or HEXADECIMAL.

To set the CM log level to 3:

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_COMPONENT    STR [0] "CM"
0 PIN_FLD_LOGLEVEL     INT [0] 3
```

Example 1–323 Sample Output Flist

This is a sample output flist returned when you set the CM log level to 3:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_LOGLEVEL        INT [0] 3
```

PCM_OP_INFMGR_START_SERVER

Starts servers running on a node or in a cell.

If the object type is **CELL**, the output includes one PIN_FLD_CELLS for each cell. In each PIN_FLD_CELLS is one or more PIN_FLD_NODES.

If the object type is **NODE** or **SERVER**, the output is one or more PIN_FLD_NODES fields.

PCM_OP_INFMGR_STOP_SERVER

Stops a server.

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

Universal Message Store FM Standard Opcodes

The opcodes in Table 1–88 support Universal Message Store functionality.

Header File

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

Opcode Index

Table 1–88 Universal Message Store FM Standard Opcodes

Opcode	Description	Use
PCM_OP_UMS_GET_MESSAGE	Retrieves **/message** objects. See the discussion on retrieving message objects in the consumer application in *BRM Developer's Guide*.	Recommended
PCM_OP_UMS_GET_MESSAGE_TEMPLATE	Retrieves message templates from **/strings** objects. See the discussion on retrieving message templates in *BRM Developer's Guide*.	Recommended
PCM_OP_UMS_GET_MESSAGE_TEMPLATES	Retrieves a list of message templates for the current brand. See the discussion on retrieving message templates in *BRM Developer's Guide*.	Recommended
PCM_OP_UMS_SET_MESSAGE	Creates a **/message** object from the message template, filling in placeholders with supplied data. See the discussion on creating message objects in *BRM Developer's Guide*.	Recommended

PCM_OP_UMS_GET_MESSAGE

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.

See the discussion on retrieving message objects in the consumer application in *BRM Developer's Guide*.

Example 1–324 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /bill -1 0
0 PIN_FLD_BILL_OBJ       POID [0] 0.0.0.1 /bill 8747 5
```

Example 1–325 Sample Output Flist

```
0 PIN_FLD_POID            POID [0] 0.0.0.1 /bill 8725 5
0 PIN_FLD_RESULTS        ARRAY [0] allocated 20, used 4
1     PIN_FLD_LOCALE          STR [0] "en_US"
1     PIN_FLD_TEMPLATE_NAME   STR [0] "First Reminder"
1     PIN_FLD_DOMAIN          STR [0] "Messages - invoice reminder"
1     PIN_FLD_STRING          STR [0] "Your account is now past due in the amount
of 89.85,
                                      which was due on 03/23/03. Please send in
your payment.
```

PCM_OP_UMS_GET_MESSAGE_TEMPLATE

Retrieves message templates from **/strings** objects. Applications that produce messages for the UMS framework call this opcode to retrieve message templates.

See the discussion on retrieving message templates in *BRM Developer's Guide*.

Example 1–326 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /strings 8750 0
```

Example 1–327 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /strings 8750 0
0 PIN_FLD_LOCALE          STR [0] "en_US"
0 PIN_FLD_TEMPLATE        STR [0] "Your account is now past due in
                                  the amount of %1, which was due
                                  on %2, please send in your
                                  payment promptly."
0 PIN_FLD_TEMPLATE_NAME   STR [0] "First Reminder"
0 PIN_FLD_DOMAIN          STR [0] "Messages - invoice reminder"
```

PCM_OP_UMS_GET_MESSAGE_TEMPLATES

Retrieves a list of message templates for the current brand.

See the discussion on retrieving message templates in *BRM Developer's Guide*.

Example 1–328 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account -1 0
0 PIN_FLD_LOCALE          STR [0] "en_US"
0 PIN_FLD_DOMAIN          STR [0] "Messages - invoice reminder"
```

Example 1–329 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /search -1 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 3, used 3
1     PIN_FLD_POID           POID [0] 0.0.0.1 /strings 9422 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_HELP_STRING     STR [0] "First Reminder"
```

PCM_OP_UMS_SET_MESSAGE

Creates **/strings** objects. This opcode retrieves the message template you specify and replaces any placeholders with data specified in the input flist.

See the discussion on creating message objects in *BRM Developer's Guide*.

Example 1–330 Sample Input Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /strings 9422 0
0 PIN_FLD_LOCALE          STR [0] "en_US"
0 PIN_FLD_BILL_OBJ       POID [0] 0.0.0.1 /bill 16096 0
0 PIN_FLD_ARGS          ARRAY [1] allocated 20, used 1
1     PIN_FLD_VALUE           STR [0] "3748.06"
0 PIN_FLD_ARGS          ARRAY [2] allocated 20, used 1
1     PIN_FLD_VALUE           STR [0] "11/03/03"
```

Example 1–331 Sample Output Flist

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /message 14360 0
```

Voucher Manager FM Policy Opcodes

Use the opcodes in Table 1–89 to customize how vouchers are created and managed.

Header File

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

Opcode Index

Table 1–89 Voucher Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE	Calculates the balance impacts of associating a voucher device with an account or a service. See the discussion on customizing voucher association in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_DEVICE_CREATE	During device creation, validates the details in the input flist. See the discussion on customizing voucher creation in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_DEVICE_SET_ATTR	During device update, ensures that the voucher card number (PIN_FLD_DEVICE_ID) cannot be changed. See the discussion on customizing voucher/service association in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_DEVICE_SET_BRAND	When changing a voucher card brand, validates that the voucher's device state is **New**. See the discussion on setting the brand for a voucher in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_ORDER_ASSOCIATE	Ensures that the sub-order cannot be associated and disassociated with the master order when the order state is not **New**. See the discussion on customizing order association in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_ORDER_CREATE	Validates the information in the input flist before an **/order/voucher** object is created. See the discussion on customizing order creation in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_ORDER_DELETE	Ensures that an order cannot be deleted when the order is in the **Received** or **Partial Receive** state. See the discussion on deleting orders in *BRM Telco Integration*.	Recommended

Table 1–89 (Cont.) Voucher Manager FM Policy Opcodes

Opcode	Description	Use
PCM_OP_VOUCHER_POL_ORDER_PROCESS	Terminates the processing of the order if the order state is **Cancel**. See the discussion on canceling orders in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_ORDER_SET_ATTR	Validates the new values passed into the input flist before an **/order/voucher** object is modified. See the discussion on customizing order attributes in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_POL_ORDER_SET_BRAND	Ensures that the brand of an order cannot be changed when the order state is **Request** or **Partial Receive**. See the discussion on setting the brand for an order in *BRM Telco Integration*.	Recommended

PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE

Calculates the balance impacts of associating a voucher device (**/device/voucher** object) with an account or a service.

This opcode is called by the PCM_OP_DEVICE_POL_ASSOCIATE opcode.

See the discussion on customizing voucher association in *BRM Telco Integration*.

Example 1–332 Sample Input Flist

```
0 PIN_FLD_POID                   POID [0] 0.0.0.1 /device/voucher 8317 0
0 PIN_FLD_PROGRAM_NAME           STR  [0] "testnap"
0 PIN_FLD_SERVICES               ARRAY [0]
1    PIN_FLD_ACCOUNT_OBJ POID    [0] 0.0.0.1 /account 10803 0
1    PIN_FLD_SERVICE_OBJ POID    [0] 0.0.0.1 /service/ip 2034 0
0 PIN_FLD_DEVICE_VOUCHER         SUBSTRUCT [0]
1    PIN_FLD_DEVICE_ID           STR  [0] "HBT002PT02151"
1    PIN_FLD_VOUCHER_PIN         STR  [0] "3777"
```

PCM_OP_VOUCHER_POL_DEVICE_CREATE

Validates a device by checking the input flist. For example, this policy opcode verifies that the numbers have the correct number of digits and use the proper syntax. It also verifies that the voucher does not already exist in the database.

You can customize this opcode to change the validation rules for creating voucher devices.

This opcode is called by the PCM_OP_DEVICE_POL_CREATE opcode when creating a voucher device.

See the discussion on customizing voucher creation in *BRM Telco Integration*.

Example 1–333 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] /device/voucher M
0 PIN_FLD_DEVICE_ID       STR [0] M
0 PIN_FLD_DEVICE_VOUCHER  SUBSTRUCT O
1 PIN_FLD_VOUCHER_PIN      STR [0] O
```

PCM_OP_VOUCHER_POL_DEVICE_SET_ATTR

Ensures that the device ID (voucher PIN) cannot be changed. Validates the deal object available in the database if the deal object is changed.

You can customize this opcode to change how vouchers are associated with services.

This opcode is called by the PCM_OP_DEVICE_POL_SET_ATTR opcode when updating a voucher card device.

See the discussion on customizing voucher/service association in *BRM Telco Integration*.

Example 1–334 Sample Input Flist

```
0 PIN_FLD_POID            POID [0] /device/voucher  M
0 PIN_FLD_DEVICE_ID       STR [0]  O
0 PIN_FLD_DEVICE_VOUCHER  SUBSTRUCT [0]  M
1 PIN_FLD_VOUCHER_PIN     STR [0]  O
1 PIN_FLD_DEAL_OBJ         POID [0]  O
```

PCM_OP_VOUCHER_POL_DEVICE_SET_BRAND

Validates that the voucher device state is **New**, when changing the voucher brand.

You can customize this opcode to change how vouchers can be associated with brands.

This opcode is called by the PCM_OP_DEVICE_POL_SET_BRAND opcode.

See the discussion on setting the brand for a voucher in *BRM Telco Integration*.

Example 1–335 Sample Input Flist

```
0  PIN_FLD_POID         POID [0] /device/voucher [M]
0  PIN_FLD_ACCOUNT_OBJ  POID [M]
```

PCM_OP_VOUCHER_POL_ORDER_ASSOCIATE

Ensures that a sub-order cannot be associated or disassociated with the master order when the order state is not **New**.

You can customize this opcode to change any validation for voucher order association.

This opcode is called by the PCM_OP_ORDER_POL_ASSOCIATE opcode.

See the discussion on customizing order association in *BRM Telco Integration*.

Example 1–336 Sample Input Flist

```
0 PIN_FLD_POID              POID[0]poid for the order object M
0 PIN_FLD_FLAGS             INT [0]association flag M
0 PIN_FLD_ORDERS            ARRAY [0]Orders array
1 PIN_FLD_ORDER_OBJ          POID [0]sub-order poid O
```

PCM_OP_VOUCHER_POL_ORDER_CREATE

Validates the information in the input flist before an order object is created.

You can customize this opcode to change the validation rules for creating **/order/voucher** objects.

This opcode is called by the PCM_OP_ORDER_POL_CREATE policy opcode.

See the discussion on customizing order creation in *BRM Telco Integration*.

Example 1–337 Sample Input Flist

```
0 PIN_FLD_POID                POID[0]poid for the order object M
0 PIN_FLD_VOUCHER_ORDERS      ARRAY[0] array of vouchers
1 PIN_FLD_START_SERIAL_NO     STR [0]Start serial number M
1 PIN_FLD_QUANTITY            DECIMAL[0]# of cards requested M
1 PIN_FLD_BATCH_TOTAL          DECIMAL [0]Total number of batches
                              in order M
1 PIN_FLD_BATCH_QUANTITY      DECIMAL [0]batch quantity M
1 PIN_FLD_PACK_QUANTITY       DECIMAL [0]pack quantity M
1 PIN_FLD_DEAL_OBJ             POID [0]Deal Object M
```

PCM_OP_VOUCHER_POL_ORDER_DELETE

Ensures that an order cannot be deleted when the order is in the **Received** or **Partial Receive** state.

This opcode is called by the PCM_OP_ORDER_POL_DELETE opcode.

See the discussion on deleting orders in *BRM Telco Integration*.

Example 1–338 Sample Input Flist

```
0 PIN_FLD_POID           POID[0]poid for the order object  M
```

PCM_OP_VOUCHER_POL_ORDER_PROCESS

Reads the status of an order using the order POID and terminates the processing of the order if the order state is **Cancel**.

This opcode is called by the PCM_OP_ORDER_POL_PROCESS opcode.

See the discussion on canceling orders in *BRM Telco Integration*.

Example 1–339 Sample Input Flist

```
0 PIN_FLD_POID             POID [0] /order/voucher M
0 PIN_FLD_DUPLICATE        INT [0] Duplicate entries
0 PIN_FLD_COUNT            INT [0] Devices quantity
0 PIN_FLD_QUANTITY         DECIMAL [0] Total Quantity ordered
0 PIN_FLD_QUANTITY_APPLIED  DECIMAL [0] Quantity processed  0
```

PCM_OP_VOUCHER_POL_ORDER_SET_ATTR

Validates the new values passed into the input flist before an order is modified.

This opcode is called by the PCM_OP_ORDER_POL_SET_ATTR opcode.

See the discussion on customizing order attributes in *BRM Telco Integration*.

Example 1–340 Sample Input Flist

```
0 PIN_FLD_POID            POID[0]poid for the order object M
0 PIN_FLD_RESULTS         ARRAY[0] response file info O
0 PIN_FLD_FILES           ARRAY[0]request file info O
0 PIN_FLD_VOUCHER_ORDERS  ARRAY[0] array of vouchers
1 PIN_FLD_START_SERIAL_NO STR[0]starting serial number O
1 PIN_FLD_QUANTITY        DECIMAL[0]# of cards requested O
1 PIN_FLD_BATCH_TOTAL      DECIMAL [0]Total number of batches in
                          order O
1 PIN_FLD_BATCH_QUANTITY  DECIMAL [0]batch quantity O
1 PIN_FLD_PACK_QUANTITY   DECIMAL [0]pack quantity O
1 PIN_FLD_DEAL_OBJ         POID [0]Deal Object O
```

PCM_OP_VOUCHER_POL_ORDER_SET_BRAND

Ensures that the brand of an order cannot be changed when the order state is in **Request** or **Partial Receive**.

This opcode is called by the PCM_OP_ORDER_POL_SET_BRAND opcode.

See the discussion on setting the brand for an order in *BRM Telco Integration*.

Example 1–341 Sample Input Flist

```
0 PIN_FLD_POIDPOID[0]poid for the order object   M
0 PIN_FLD_ACCOUNT_OBJPOID[0]poid of the brand of the order M
```

Voucher Manager FM Standard Opcodes

The opcodes listed in Table 1–90 perform voucher management.

Header File

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

Opcode Index

Table 1–90 Voucher Management FM Standard Opcodes

Opcode	Description	Use
PCM_OP_VOUCHER_ASSOCIATE_VOUCHER	Initiates operations that calculate the balance impacts of the deal linked to a voucher and that associate the voucher with an account or a service. See the discussion on how voucher association works in *BRM Telco Integration*.	Recommended
PCM_OP_VOUCHER_EXPIRATION	Changes the devices state.	Recommended

PCM_OP_VOUCHER_ASSOCIATE_VOUCHER

Calls PCM_OP_DEVICE_ASSOCIATE to perform these operations:

- Calculate the balance impacts of purchasing the deal linked to the voucher device (**/device/voucher** object).
- Associate a voucher device with an account or a service.

See the discussion on how voucher association works in *BRM Telco Integration*.

PCM_OP_VOUCHER_EXPIRATION

Performs these operations:

- Searches for the device POIDs that are in New state (**1**).
- Calls PCM_OP_DEVICE_SET_STATE for each device and changes the device state to Expired (**3**). You must configure this state transition in the **pin_device_state_voucher** file.

Zone Map FM Policy Opcodes

The opcodes in Table 1–91 provide BRM with the support for rating the zones that you create in the Zone Mapper.

Header File

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

Opcode Index

Table 1–91 Zone Map FM Policy Opcodes

Opcode	Description	Use
PCM_OP_ZONEMAP_POL_GET_LINEAGE	Searches a given zone map for data associated with a given string. See the discussion on finding zone maps in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ZONEMAP_POL_GET_ZONEMAP	Retrieves pricing zone map data from the BRM database. See the discussion on getting zone maps from the BRM database in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ZONEMAP_POL_SET_ZONEMAP	Saves pricing zone map information to the BRM database. See the discussion on saving zone map data in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_ZONEMAP_POL_GET_LINEAGE

Searches a given zone map for the data associated with a given string. You supply a string and a zone map name. It then searches the zone map for the given string and returns the matching node with all ancestors of the matching node (the *lineage*).

This opcode is called by internal rating opcodes.

See the discussion on finding zone maps in *BRM Setting Up Pricing and Rating*.

PCM_OP_ZONEMAP_POL_GET_ZONEMAP

Retrieves zone map data from the BRM database and displays zone maps in the Zone Mapper.

This opcode is called by the PCM_OP_ZONEMAP_GET_CAAR_MATRIX standard opcode.

See the discussion on getting zone maps from the BRM database in *BRM Setting Up Pricing and Rating*.

PCM_OP_ZONEMAP_POL_SET_ZONEMAP

Saves zone map in the BRM database when you commit zone maps in the Zone Mapper. BRM stores this information is the **/zonemap** object.

This opcode is called by the PCM_OP_ZONEMAP_COMMIT_ZONEMAP standard opcode.

See the discussion on saving zone map data in *BRM Setting Up Pricing and Rating*.

Zone Map FM Standard Opcodes

The opcodes listed in Table 1–92 provide BRM with the support for rating the zones that you create in the Zone Mapper.

Header File

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

Opcode Index

The next table contains the list of the standard zone map opcodes.

Table 1–92 Zone Map FM Standard Opcodes

Opcode	Description	Use
PCM_OP_ZONEMAP_COMMIT_ZONEMAP	Commits zone map changes to the BRM database (possible operations include deletion, creation, and updates of zone maps). See the discussion on how zone mapping works in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_ZONEMAP_GET_ZONEMAP	Retrieves zone map data from the BRM database. See the discussion on how zone mapping works in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_ZONEMAP_COMMIT_ZONEMAP

Commits zone map changes to the BRM database. You can add, update, or delete a zone map.

See the discussion on how zone mapping works in *BRM Setting Up Pricing and Rating*.

PCM_OP_ZONEMAP_GET_ZONEMAP

Retrieves zone maps from the BRM database.

See the discussion on how zone mapping works in *BRM Setting Up Pricing and Rating*.

2

PIN Libraries Reference

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Portal Information Network (PIN) libraries.

Configuration File-Reading Functions

Use these functions to read configuration files, such as **pin.conf** files.

pin_conf

This BRM library routine reads a single configuration value from a configuration file.

The Connection Manager (CM), Data Manager (DM), and Portal Communications Module (PCM) libraries all use this routine to read the configuration information.

When first called, this routine looks for the configuration file specific to the application. See "Locations of Configuration and Properties Files" in *BRM System Administrator's Guide*. The library returns an error if it cannot locate the configuration file.

This routine uses regular **malloc**. If you are using this routine in a Storage Manager to get data to put on an flist, use **SET** (*not* **PUT**), and then free the storable object by using the regular **free** routine when you are finished.

> **Important:** Do not use this routine if performance is a consideration and you use the routine often.

For more information on configuration files, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

For information on reading multiple configuration values from a file, see "pin_conf_multi".

Syntax

```
#include "pcm.h"
void
pin_conf(
                char          *prog_name,
                char          *token,
                int32          valtype,
                caddr_t*     **valpp,
                int32         *errp);
```

Parameters

prog_name
The program name this routine looks for in the configuration file. If *prog_name* is **NULL**, the routine looks only for entries marked with a program of "-". If *prog_name* is any other value, the routine looks for either a specific match or "-" in the program parameter. For a description of configuration file syntax, see "Configuration Entry Syntax" in *BRM System Administrator's Guide*.

token
The name of the configuration entry keyword this routine looks for in the configuration file.

valtype
The **type** of the value the routine reads in the configuration entry. This parameter tells the routine how to interpret the entry value. The supported types are:

- **PIN_FLDT_INT**
- **PIN_FLDT_DECIMAL**
- **PIN_FLDT_STR**

- **PIN_FLDT_POID**

valpp
The **ptr-ptr** used to pass back the location of the value for the entry. The memory for the value is dynamically allocated, and the filled-in pointer **type** matches the value **type**.

errp
A pointer to the error buffer, which passes error information back to the caller.

Return Values

This routine returns nothing.

This routine passes error status back to the caller. If it finds a matching entry in the configuration file, it passes back **PIN_ERR_NONE**. If it does not find a matching entry, it passes back **PIN_ERR_NOT_FOUND**. The routine might also pass back other error values.

pin_conf_beid

This library routine reads values for BRM resources from the **/config/beid** storable object.

Syntax

```
#include "pin_errs.h"
#include "pcm.h"
pin_flist_t*
pin_conf_beid(
               pcm_context_t     *ctxp,
               pin_errbuf_t      *ebufp);
```

Parameters

ctxp
A pointer to an open context. This routine gets the database number from the configuration file of the current application and queries that database for the **/config/beid** object.

ebufp
A pointer to the error buffer, which passes error information back to the caller.

Return Values

Returns values for the **/config/beid** storable object data as an flist.

Error Handling

This routine sets the return flist to **NULL** and provides more information about the error in the error buffer if there is an error.

pin_conf_multi

This library routine reads multiple configuration values of the same type from a configuration file. To do this, you reuse this routine until it returns **PIN_ERR_NOT_FOUND**. This routine uses the **time_t** value to monitor the configuration file for changes throughout this operation and returns an error if the state of the file changes.

The Connection Manager (CM), Data Manager (DM), and PCM libraries all use this routine to read the configuration information.

When first called, this routine looks for the configuration file specific to the application. See "Locations of Configuration and Properties Files" in *BRM System Administrator's Guide*. The library returns an error if it cannot locate the configuration file.

This routine uses regular **malloc**. If you are using this routine in a Storage Manager to get data to put on an flist, use **SET** (*not* **PUT**), and then free the storable object by using the regular **free** routine when you are finished.

Important: Do not use this routine if performance is a consideration and you use the routine often.

For more information on configuration files, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

For information on reading a single configuration value from a file, see "pin_conf".

Syntax

```
#include "pcm.h"
void
pin_conf(
                char          *prog_name,
                char          *token,
                int32          valtype,
                caddr_t*     **valpp,
                int32         *linep,
                time_t        *modtp,
                int32         *errp);
```

Parameters

prog_name
The program name this routine looks for in the configuration file. If *prog_name* is **NULL**, the routine looks only for entries marked with a program of "-". If *prog_name* is any other value, the routine looks for either a specific match or "-" in the program parameter. For a description of configuration file syntax, see "Configuration Entry Syntax" in *BRM System Administrator's Guide*.

token
The name of the configuration entry keyword this routine looks for in the configuration file.

valtype
The **type** of the value the routine reads in the configuration entry. This parameter tells the routine how to interpret the entry value. The supported types are:

- **PIN_FLDT_INT**
- **PIN_FLDT_DECIMAL**
- **PIN_FLDT_STR**
- **PIN_FLDT_POID**

valpp
The **ptr-ptr** used to pass back the location of the value for the entry. The memory for the value is dynamically allocated, and the filled-in pointer **type** matches the value **type**.

linep
A pointer to a line number. Passes an integer back to the caller to identify the line where the last value was found. Initialize to zero on the first call.

modtp
A pointer to a time variable. Passes a timestamp back to the caller to compare to the last timestamp. Initialize to zero on the first call.

errp
A pointer to the error status, which passes error information back to the caller.

Return Values

This routine returns nothing.

This routine passes error status back to the caller.

- If it finds a matching entry in the configuration file, it passes back **PIN_ERR_NONE**. This indicates that the routine then reuses the key to look for another matching entry (as long as it has not generated a PIN_ERR_STALE_CONF error).
- If it does not find a matching entry, it passes back **PIN_ERR_NOT_FOUND**. This signals the end of the routine.
- If it detects, based on a change in the **time_t** value, that the configuration file has been opened, modified, or has otherwise changed since it first accessed the file (jeopardizing the ability of the routine to maintain correct reference to the last value read), it passes back **PIN_ERR_STALE_CONF**.

> **Important:** In this case, you must restart the entire process.

The routine may also pass back other error values.

Decimal Data Type Manipulation Functions

This section describes decimal data type manipulation functions.

About Using the API

The decimal data type application programming interface (API) consists of a minimal set of methods that provides all the functionality you need to perform basic mathematical functions, comparison, and format conversion with the decimal data type. Input and output to the functions are provided using number strings or floating point doubles.

> **Tip:** Use strings to avoid small quantity errors; for example, 31.299999999 vs. 31.3.

If there are errors, functions that return a **pin_decimal_t** return **NULL**. **pbo_decimal_destroy** allows **NULL**.

International Platform Issues

The **pin_decimal** function expects the decimal point character to be that of the locale. For US systems, this is a period; for most international platforms, it is a comma.

> **Caution:** Do not pass a string with a hard-coded decimal point to **::pin_decimal** because **pin_decimal** will return a NULL pointer in platforms that do not use a period for the decimal point character.

About Rounding Modes

This section defines the rounding modes that you pass as input parameters in the following functions:

- pbo_decimal_round
- pbo_decimal_round_assign
- pbo_decimal_from_double
- pbo_decimal_from_double_round

The rounding modes in Table 2–1 are defined in **pcm.h**. They have the same names and functionality as the Java BigDecimal Datatype.

Table 2–1 Rounding Modes

Rounding Mode	Description
ROUND_UP	Rounds up to the nearest number of the appropriate scale. **Examples**: 21.11 rounds to 21.2 when the scale is one decimal place.
ROUND_DOWN	Rounds down to the nearest number of the appropriate scale. **Examples**: 21.19 rounds to 21.1 when the scale is one decimal place.

Table 2–1 (Cont.) Rounding Modes

Rounding Mode	Description
ROUND_DOWN_ALT	Rounds down after first rounding to the nearest using a scale of two more than the one configured. This method compensates for possible loss of precision when numbers are rounded down during certain computations, such as when prorating cycle fees. For more information, see "About Rounding Modes That Correct for Loss of Precision" in *BRM Setting Up Pricing and Rating*.
ROUND_CEILING	If the number is positive, rounding is the same as for ROUND_UP; if negative, the same as for ROUND_DOWN.
ROUND_FLOOR	If the number is positive, rounding is the same as for ROUND_DOWN; if negative the same as for ROUND_UP. This method allows you to round to benefit customers. For example, if rounding is set to two significant digits, a credit to a customer of -7.999 is rounded to -8.00, and a debit of 7.999 is rounded to 7.99.
ROUND_FLOOR_ALT	Rounds using ROUND_FLOOR after first rounding to the nearest using a scale of two more than the one configured. This method compensates for possible loss of precision when numbers are rounded down during certain computations, such as when prorating cycle fees. For more information, see "About Rounding Modes That Correct for Loss of Precision" in *BRM Setting Up Pricing and Rating*.
ROUND_HALF_UP	If the discard part is .5 or higher round up; otherwise, round down. **Examples**: 21.15 rounds to 21.2, 21.14 rounds to 21.1, etc. This is the most common rounding method.
ROUND_HALF_DOWN	If the discard part is more than .5, round up; if it is .5 or less, round down. **Examples**: 21.16 rounds to 21.2, 21.15 rounds to 21.1.
ROUND_HALF_EVEN	If the digit to the left of the discard is odd, rounding is the same as for ROUND_HALF_UP. If the digit to the left is even, rounding is the same as for ROUND_HALF_DOWN. **Examples:** 1.049 rounds to 1.0 1.050 rounds to 1.0 1.051 rounds to 1.1 1.149 rounds to 1.1 1.150 rounds to 1.2 1.151 rounds to 1.2
ROUND_UNNECESSARY	Rounding not allowed. If rounding is attempted with this rounding mode, an error is returned.

About Scaling

A decimal data type is based on the Java BigDecimal data type. It is an immutable, arbitrary-precision signed decimal number, which consists of an arbitrary precision integer value and a non-negative integer scale, which represents the number of decimal digits to the right of the decimal point.

For this implementation, the scale is set at 15, meaning numbers carry up to 15 decimal places. For operations that would normally result in a value with a larger scale, the value is rounded to 15 decimal places. For example, when multiplying the two decimal data types 12.528694120521357 and 4.126943650923412, the mathematical result would normally be 51.705214655047095455751917310084, which has a scale of 30. However, because the scale is set at 15, the product is rounded to 51.705214655047095 and a consistent scale of 15 is maintained.

About Memory Management

For functions that allocate memory for the **pin_decimal_t** structure, make sure that the memory is reclaimed after the **pin_decimal_t** is no longer needed. If **pin_decimal_t** has been passed to an flist with PIN_FLIST_PUT, use **pin_flist_destroy** to reclaim memory. Otherwise, use **pbo_decimal_destroy**.

assign functions do not allocate new memory; instead, they replace the first parameter with the new value. Therefore, there is no need to reclaim memory.

pbo_decimal_abs

This function returns a pointer to a newly allocated **pin_decimal_t**, which is the absolute value of the input **pin_decimal_t**.

Syntax

```
pin_decimal_t*
pbo_decimal_abs(
               const pin_decimal_t     *pdp,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_abs_assign

This function replaces the input **pin_decimal_t** with its absolute value.

Syntax

```
pin_decimal_t*
pbo_decimal_abs_assign(
               pin_decimal_t     *pdp,
               pin_errbuf_t      *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_add

This function adds the two decimals passed in and returns a pointer to a newly allocated **pin_decimal_t**. The scale of the output is the larger of the scales of the two inputs.

Syntax

```
pin_decimal_t*
pbo_decimal_add(
               const pin_decimal_t     *pdp1,
               const pin_decimal_t     *pdp2,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp1
A pointer to the input **pin_decimal_t**.

pdp2
A pointer to another input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_add_assign

This function replaces the value of the first **pin_decimal_t** with the sum of itself and another **pin_decimal_t**.

Syntax

```
void
pbo_decimal_add_assign(
               pin_decimal_t           *pdp1,
               const pin_decimal_t     *pdp2,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_compare

This function compares the first input decimal with the second input decimal and returns one of the following values to indicate the difference between the input decimals:

- **-1** if *pdp1* < *pdp2*
- **0** if *pdp1* = *pdp2*

> **Note:** *pdp1* is considered equal to *pdp2* if the difference between them is less than 10^{-12}.

- **1** if *pdp1* > *pdp2*
- **0** in the event of an error.

Syntax

```
int
pbo_decimal_compare(
                const pin_decimal_t     *pdp1,
                const pin_decimal_t     *pdp2,
                pin_errbuf_t            *ebufp);
```

Parameters

pdp1
A pointer to the first **pin_decimal_t**.

pdp2
A pointer to the second **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_copy

This function makes a copy of the input **pin_decimal_t** and returns a pointer to the newly allocated **pin_decimal_t**.

Syntax

```
pin_decimal_t*
pbo_decimal_copy(
               const pin_decimal_t     *pdp,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_destroy

This function frees all the memory associated with the specified **pin_decimal_t** and sets **decpp* to NULL.

Syntax

```
void
pbo_decimal_destroy(
                pin_decimal_t     **decpp);
```

Parameter

decpp
A pointer to a pointer to the **pin_decimal_t** to be deleted. Can be set to **NULL** (the function does nothing).

pbo_decimal_divide

This function divides the first input parameter by the second input parameter and returns a pointer to a newly allocated **pin_decimal_t**.

> **Note:** Rounding is performed according to preset rounding and scaling. The default rounding mode is ROUND_DOWN and the scaling is set at 15 decimal places.

Syntax

```
pin_decimal_t*
pbo_decimal_divide(
               const pin_decimal_t     *num,
               const pin_decimal_t     *byp,
               pin_errbuf_t            *ebufp);
```

Parameters

num
A pointer to the dividend.

byp
A pointer to the divisor.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_BAD_ARG if one of the following is true:
  - The scale is less than 0.
  - The rounding mode is unknown.
  - Either the dividend or the divisor is not a valid **pin_decimal_t**.
  - An attempt was made to divide by 0.
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_divide_assign

This function divides the dividend by the divisor and stores the result in the dividend.

Syntax

```
void
pbo_decimal_divide_assign(
                pin_decimal_t           *nump,
                const pin_decimal_t     *byp,
                pin_errbuf_t            *ebufp);
```

Parameters

nump
A pointer to the dividend.

byp
A pointer to the divisor.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_BAD_ARG if one of the following is true:
  - The scale is less than 0.
  - The rounding mode is unknown.
  - Either the dividend or the divisor is not a valid **pin_decimal_t**.
  - An attempt was made to divide by 0.
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_from_double

This function constructs a **pin_decimal_t** data type from the double-precision floating point number (allocates memory) and returns a pointer to the newly created **pin_decimal_t** data type.

> **Note:** Because of the inherent rounding errors associated with converting a double to a decimal data type, you should avoid using this function whenever possible. Use **pbo_decimal_from_str** instead. If you must use doubles, use the **pbo_decimal_from_double_round** function.

Syntax

```
pin_decimal_t
*pbo_decimal_from_double(
                double            d,
                pin_errbuf_t     *ebufp);
```

Parameters

d
The input of type double float (a double-precision floating point number).

ebufp
A pointer to the error buffer.

See also "pbo_decimal_from_str".

pbo_decimal_from_double_round

This function provides an option for choosing the rounding mode. (See "About Rounding Modes".)

Constructs a **pin_decimal_t** data type from the double-precision floating point number (allocates memory) and returns a pointer to the newly created **pin_decimal_t** data type.

> **Note:** Because of the inherent rounding errors associated with converting a double to a decimal data type, you should avoid using this function whenever possible. Use **pbo_decimal_from_str** instead.

Syntax

```
pin_decimal_t*
pbo_decimal_from_double_round(
                double            value,
                int               rounding_mode,
                pin_errbuf_t     *ebufp)
```

Parameters

value
The value to convert.

rounding_mode
See "About Rounding Modes".

ebufp
A pointer to the error buffer.

pbo_decimal_from_str

This function constructs a **pin_decimal_t** data type from an input string and returns a pointer to the newly created **pin_decimal_t** data type.

This function understands **NULL** to create a NULL-valued **pin_decimal_t**. The string does not need to end with a null character, but parsing will end at either a null character or any white space character.

This function ignores leading spaces, tabs, and leading 0's and checks on non-numeric types.

This function detects the sign (+ or -) and stores it. This function accepts the same input at **strtod** except that an exponent is not allowed, and only base 10 is supported.

Syntax

```
pin_decimal_t*
pbo_decimal_from_str(
               const            *str,
               pin_errbuf_t     *ebufp);
```

Parameters

str
The input number string.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the string pointer is **NULL**
- PIN_ERR_BAD_ARG if there were multiple decimal points before null or space or if it cannot derive a valid number from the string
- PIN_ERR_NO_MEM if the function cannot allocate memory for **pbo_decimal**

pbo_decimal_is_null

This function verifies if the input **pin_decimal_t** is **NULL**.

Syntax

```
int
pbo_decimal_is_null(
                const pin_decimal_t     *pdp,
                pin_errbuf_t            *ebufp);
```

Parameters

pdp
The pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns PIN_ERR_BAD_ARG indicating that a non-NULL pointer points to a data area not marked as a valid **pin_decimal_t**.

pbo_decimal_is_zero

This function checks if the input value is a valid **pin_decimal_t** and has a zero value. Returns **1** if the conditions are met; otherwise, it returns **0**.

Syntax

```
int
pbo_decimal_is_zero(
               const pin_decimal_t     *pdp,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns PIN_ERR_BAD_ARG indicating that a non-NULL pointer points to a data area that is not marked as a valid **pin_decimal_t**.

pbo_decimal_multiply

This function multiplies the two input **pin_decimal_t** values and returns a pointer to a new **pin_decimal_t** that is the product.

Syntax

```
pin_decimal_t*
pbo_decimal_multiply(
               const pin_decimal_t     *pdp1,
               const pin_decimal_t     *pdp2,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp1
The pointer to an input **pin_decimal_t**.

pdp2
The pointer to another input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_multiply_assign

This function multiplies two **pin_decimal_t** data types and stores the product in the first **pin_decimal_t**.

For example, if **a=**10 and **b=**2, after calling **pbo_decimal_multiply_assign(a, b, *ebufp)**, **a** is equal to 20.

Syntax

```
void
pbo_decimal_multiply_assign(
               pin_decimal_t           *pdp1,
               const pin_decimal_t     *pdp2,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp1
The pointer to an input **pin_decimal_t**.

pdp2
The pointer to another input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued

pbo_decimal_negate

This function returns a pointer to a new **pin_decimal_t** that has the reverse sign of the input decimal. If the input decimal has a value of **0**, it returns a pointer to another **pin_decimal_t** with the value of **0**.

Table 2–2 contains examples, where *x* is a pointer **pin_decimal_t**:

Table 2–2 pbo_decimal_negate Examples

Value to Which x Points	pbo_decimal_negate(x, ebuf) Returns a New Pointer to This Value:
5	-5
0	0
-3	3
Value to which x points	pbo_decimal_negate(x, ebuf) returns a new pointer to a value of:

Syntax

```
pin_decimal_t*
pbo_decimal_negate(
                const pin_decimal_t     *pdp,
                pin_errbuf_t            *ebufp);
```

Parameters

pdp
The pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_negate_assign

This function reverses the sign of the input **pin_decimal_t**.

Syntax

```
pin_decimal_t*
pbo_decimal_negate_assign(
               pin_decimal_t     *pdp,
               pin_errbuf_t      *ebufp);
```

Parameters

pdp
The pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued

pbo_decimal_round

This function returns a pointer to a new **pin_decimal_t** that contains the value of the first argument rounded according to the specified scale and rounding mode.

Syntax

```
pin_decimal_t*
pbo_decimal_round(
                const pin_decimal_t     *decp,
                int32                    scale,
                int32                    rounding_mode,
                pin_errbuf_t            *ebufp);
```

Parameters

decp
A pointer to the input **pin_decimal_t**.

scale
See "About Scaling".

rounding_mode
See "About Rounding Modes".

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_round_assign

This function replaces the value of the first argument with the value of the argument rounded according to the specified scale and rounding mode.

Syntax

```
void
pbo_decimal_round_assign(
                pin_decimal_t     *decp,
                int32              scale,
                int32              rounding_mode,
                pin_errbuf_t      *ebufp);
```

Parameters

decp
A pointer to the input **pin_decimal_t**.

scale
See "About Scaling".

rounding_mode
See "About Rounding Modes".

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is NULL
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_BAD_ARG if *decp* is an invalid value

pbo_decimal_sign

This function returns the sign of the **pin_decimal_t** argument: **-1** if the argument is negative, **0** if the argument is zero or if there is an error, or **1** if the argument is positive.

Syntax

```
int
pbo_decimal_sign(
                const pin_decimal_t     *pdp,
                pin_errbuf_t            *ebufp);
```

Parameters

pdp
The pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued

pbo_decimal_subtract

This function subtracts two **pin_decimal_t** parameters and returns a pointer to a new **pin_decimal_t** containing the difference.

Syntax

```
pin_decimal_t*
pbo_decimal_subtract(
               const pin_decimal_t     *nump,
               const pin_decimal_t     *byp,
               pin_errbuf_t            *ebufp);
```

Parameters

nump
The pointer to the **pin_decimal_t** from which to subtract.

byp
The pointer to the **pin_decimal_t** to subtract.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

pbo_decimal_subtract_assign

This function subtracts a decimal from another decimal and replaces the value of the first decimal with the difference.

For example, if **a=**8 and **b=**3, after calling **pbo_decimal_subtract_assign (a, b, ebuf)**, **a** is equal to **5**.

Syntax

```
void
pbo_decimal_subtract_assign(
               pin_decimal_t           *pdp1,
               const pin_decimal_t     *pdp2,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp1
The pointer to an input **pin_decimal_t**.

pdp2
The pointer to another input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued

pbo_decimal_to_double

This function converts the input **pin_decimal_t** into a double-precision floating point number.

If **pin_decimal_t** is not **NULL**, this function converts **pin_decimal_t** to a string using **pin_decimal_to_str(NULL format,...)** and then **strtod**.

Syntax

```
double
pbo_decimal_to_double(
                const pin_decimal_t     *pdp,
                pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**
- PIN_ERR_BAD_ARG if **strtod** returns an error

See also **pin_decimal_to_str().**

pbo_decimal_to_str

This function creates an ASCII string representation of the input decimal value.

If successful, the function returns a pointer to the allocated null-terminated string. If there are errors, it returns **NULL**.

Syntax

```
char*
pbo_decimal_to_str(
               const pin_decimal_t     *pdp,
               pin_errbuf_t            *ebufp);
```

Parameters

pdp
A pointer to the input **pin_decimal_t**.

ebufp
A pointer to the error buffer.

Error Handling

If there are errors, this function returns the following error status:

- PIN_ERR_NULL_PTR if the input **pin_decimal_t** pointer is **NULL**
- PIN_ERR_IS_NULL if the input **pin_decimal_t** is **NULL**-valued
- PIN_ERR_NO_MEM if the function cannot allocate memory for the output **pin_decimal_t**

Error-Handling Macros

This section describes error-handling macros.

PIN_ERR_LOG_EBUF

This BRM macro logs a standardized message that includes details of the error condition recorded in an error buffer. It provides a convenient method for logging errors returned by API calls that use the error buffer to pass back status. The caller can specify an additional message that is appended to the standard format.

Syntax

```
#include "pcm.h"
void
PIN_ERR_LOG_EBUF(
                 int32          level,
                 char          *msg,
                 pin_errbuf_t  *ebufp);
```

Parameters

level
The level of this log message. Based on the level specified and the logging level set in the log system, the message is either printed or discarded. See "PIN_ERR_SET_LEVEL" for the error level descriptions.

msg
A string to be printed in addition to the standard logging message. Allows additional detailed information to be added to the log message by the caller.

ebufp
A pointer to the error buffer containing the error condition. The values in the error buffer are printed in human-readable form as part of the log message.

Return Values

This macro returns nothing.

Error Handling

There are no error conditions for this macro. If the message cannot be logged for any reason, that information is not passed back to the caller.

PIN_ERR_LOG_FLIST

This macro prints the contents of an flist to the error log file. It allows an application to log an arbitrary message and the corresponding flist for recording errors, accounting, or debugging. The specified message and flist are logged in the standard log entry format, so complete information about where they came from is available in the log file.

Syntax

```
#include "pcm.h"
void
PIN_ERR_LOG_FLIST(
                int32         level,
                char         *msg,
                pin_flist_t  *flistp);
```

Parameters

level
The level of this log message. Based on the level specified and the logging level set in the log system, the message is either printed or discarded. See "PIN_ERR_SET_LEVEL" for the error-level descriptions.

msg
A string to be printed in addition to the standard logging message. Allows additional detailed information to be added to the log message by the caller.

flistp
A pointer to the flist to be printed in addition to the log message.

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_ERR_LOG_MSG

This macro logs the specified message to the log file. It allows an application to log arbitrary messages for recording errors or debug information. The specified message is logged in the standard log entry format, so complete information about where the message came from is available in the log file.

Syntax

```
#include "pcm.h"
void
PIN_ERR_LOG_MSG(
                int32     level,
                char     *msg);
```

Parameters

level
The level of this log message. Based on the level specified and the logging level set in the log system, the message is either printed or discarded. See "PIN_ERR_SET_LEVEL" for the error-level descriptions.

msg
A string to be printed in addition to the standard logging message. Allows additional detailed information to be added to the log message by the caller. Special characters should be escaped if you want them to be printed without modification.

Return Values

This macro returns nothing.

Error Handling

There are no error conditions for this macro. If the message cannot be logged for any reason, that information is not passed back to the caller.

PIN_ERR_LOG_POID

This macro prints the contents of a POID to the error log file. This operation allows an application to log an arbitrary message and the corresponding POID for recording errors, accounting, or debugging. The specified message and POID are logged in the standard log entry format, so complete information about where they came from is available in the log file.

Syntax

```
#include "pcm.h"
void
PIN_ERR_LOG_POID(
                int32     level,
                char     *msg,
                poid_t   *pdp);
```

Parameters

level
The level of this log message. Based on the level specified and the logging level set in the log system, the message is either printed or discarded. See "PIN_ERR_SET_LEVEL" for the error-level descriptions.

msg
A string to be printed in addition to the standard logging message. Allows additional detailed information to be added to the log message by the caller.

pdp
A pointer to the POID to be printed in addition to the standard log entry information.

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_ERR_SET_LEVEL

This macro sets the desired level of logging. Messages sent to the logging system have a severity code that describes the category of the message. Users can chose to have messages of different categories either logged or suppressed, depending on how much logging output they would like to see. Messages that are suppressed are discarded.

In general, BRM recommends that only debug messages be suppressed on a production system. All other types of messages convey possible system problems that should be investigated. Debug messages can be enabled when they might help diagnose an application error and then suppressed when the system is running in a steady state.

If PIN_ERR_SET_LEVEL is not called, the logging system defaults to a level of 2.

Syntax

```
#include "pcm.h"
int32
PIN_ERR_SET_LEVEL(
                 int32     level);
```

Parameter

level
Sets the mask for which level of errors should be logged and which ones suppressed. All messages with a level of *level* or less are printed. All messages with a level greater than *level* are suppressed. Errors come in the levels listed in Table 2–3:

Table 2–3 PIN_ERR_SET_LEVEL Values

Allowed Level Values	System Category	Type of Message	Messages Returned
0	N/A	N/A	Nothing at this level
1	**E**	Error	Serious system integrity problems
2	**W**	Warning	Possible data corruption problems
3	**D**	Debug	Details of application errors

- Setting *level* to **0** means no messages will be produced, no matter what the error.
- Setting *level* to **1** will log only errors, which indicate some portion of the BRM system is not operating correctly.
- Setting *level* to **2** will print errors and warnings. Warnings indicate that data was found in the database that is suspect, and some data corruption may have occurred. The system can still operate properly, but specific operations related to the corrupt data may have to be bypassed.
- Setting *level* to **3** prints debug messages. The debug messages log detailed information about operations that applications attempt that generate errors in the system due to incorrect parameters or other application level errors. The system is not adversely affected by this type of event, but the application developer can use the debug messages to more easily pinpoint where the application error is located.

Return Values

Returns **0** if the macro is successful. Returns a non-zero value if an error occurred. The only possible failure is the specification of an unreasonable value for *level*.

Error Handling

Returns a non-zero value if an error occurred. In this case, the internal state of the logging system is unchanged.

PIN_ERR_SET_LOGFILE

This macro specifies the file to use for logging. The log file can be changed at any time by calling PIN_ERR_SET_LOGFILE. All messages logged after the change are logged to the new file.

If this macro is not called, the logging system uses the default **./default.pinlog** log file, where **./** is relative to the directory in which the application was started.

Syntax

```
#include "pcm.h"
int32
PIN_ERR_SET_LOGFILE(
                 char     *path);
```

Parameter

path
The path of the file to be used as the log file. The file is opened exactly as specified, so relative paths will work, but they will be relative to the current directory of the running program.

Return Values

Returns a non-zero value if an error occurred.

Error Handling

Returns a non-zero value if an error occurred. The internal state of the logging system is unchanged. The return value should be tested after the call to ensure the desired log file will be used.

PIN_ERR_SET_PROGRAM

This macro sets the program name for log messages. The program name is printed in each log message as additional information to aid in debugging problems. The program name can be set to any string desired.

If PIN_ERR_SET_PROGRAM is not called, log messages are printed with a blank program name field.

Syntax

```
#include "pcm.h"
int32
PIN_ERR_SET_PROGRAM(
                 char     *program);
```

Parameter

program
The name of the running program to be printed in log messages. If the pointer is NULL, the current name is not changed.

Return Values

Returns **0** if the macro is successful. Returns a non-zero value if an error occurred. The only possible failure condition is the specification of a NULL pointer.

Error Handling

Returns a non-zero return value if an error occurred. In this case, the internal state of the logging system is unchanged.

PIN_ERRBUF_CLEAR

This macro is used for a newly allocated or defined error buffer structure to initialize the contents of the error buffer to **0**.

Syntax

```
#include "pcm.h"
void
PIN_ERRBUF_CLEAR(
                 pin_errbuf_t     *ebufp);
```

Parameter

ebufp
A pointer to the error buffer that is initialized.

Return Values

This macro returns nothing.

Example

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

PIN_ERRBUF_IS_ERR

This macro checks the specified error buffer for an error condition. It allows an application to quickly check whether an error has occurred on a call that used the error buffer.

Macros that use individual ebuf error handling must use PIN_ERRBUF_IS_ERR after each call to test for an error.

Macros that use series-style ebuf error handling can make an entire series of calls and use this macro once at the end to test for an error.

Syntax

```
#include "pcm.h"
int32
PIN_ERRBUF_IS_ERR(
                pin_errbuf_t     *ebufp);
```

Parameter

ebufp
A pointer to an error buffer. Used by the macro to determine whether an error has occurred.

Return Values

Returns **0** if the error buffer contains no error. Returns a non-zero value if the error buffer contains an error.

Example

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

PIN_ERRBUF_RESET

This macro is called to reset the error buffer either before reusing an existing error buffer structure or before calling **pin_free** to free a dynamically allocated error buffer structure.

For details on the structure and fields in an error buffer, see "Error Buffer" in *BRM Developer's Guide*.

The use of PIN_ERRBUF_RESET depends on the type of macro called with the error buffer:

- **Individual-style ebuf:** Macros that use this style of error handling must examine the error buffer for an error after each call. Use PIN_ERRBUF_RESET to clear any error that was detected before using the same error buffer again.
- **Series-style ebuf:** Macros that use this style of error handling can use the same error buffer for a series of calls without checking for or clearing errors between calls. After a series of calls, check the error buffer for errors. Use PIN_ERRBUF_RESET to clear any error before using the error buffer again.

Syntax

```
#include "pcm.h"
void
PIN_ERRBUF_RESET(
                pin_errbuf_t     *ebufp);
```

Parameter

ebufp
A pointer to the error buffer that is reset.

Return Values

This macro returns nothing.

Example

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

pin_set_err

This function sets the error values in the **pin_errbuf_t (ebuf)** structure pointer.

> **Note:** This is the only error handling routine that is not a macro. This is a function.

Syntax

```
EXTERN
void
pin_set_err(
                pin_errbuf_t    *ebuf,
                int32            location,
                int32            pin_errclass,
                int32            pin_err,
                int32            field,
                int32            rec_ID,
                int32            reserved);
```

Parameters

ebuf
A pointer to the error buffer.

location
The location of an error. For a list of possible locations, see "BRM Error Locations" in *BRM System Administrator's Guide*.

pin_errclass
One of the four classes. See "BRM Error Classes" in *BRM System Administrator's Guide*.

pin_err
One of the system error codes. For a list of possible error codes, see "BRM Error Codes"in *BRM System Administrator's Guide*.

field
Set to **0** or to the applicable **PIN_FLD**_*xxx*.

rec_ID
Set to **0** or to the record ID of the array element the error occurred on.

reserved
Set to **0** or to a value chosen to provide further information about the specific error.

Return Values

This function returns nothing.

Error Handling

There are no error conditions for this function. If the message cannot be logged for any reason, that information is not passed back to the caller.

Flist Field-Handling Macros

This section describes flist field-handling macros.

PIN_FLIST_ANY_GET_NEXT

This BRM macro gets the value of the next simple field, substructure, or element of an array in an flist. It lets an application walk an flist retrieving each field value.

The value returned is a pointer to the actual field value, and the field remains unchanged on the original flist. The value returned must be treated as read-only to maintain the integrity of the flist. If a writable copy of the value is needed, the application must either make a copy of the returned value or take it according to its type as listed in Table 2–4:

Table 2–4 Next Field Macros

Field Type	Macro to Use
Simple	PIN_FLIST_FLD_TAKE
Substructure	PIN_FLIST_SUBSTR_TAKE
Array element	PIN_FLIST_ELEM_TAKE

Syntax

```
#include "pcm.h"
void
*PIN_FLIST_ANY_GET_NEXT(
                pin_flist_t     *flistp,
                pin_fld_num_t   *fldp,
                int32           *record_idp,
                pin_cookie_t    *cookiep,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the field being obtained.

fldp
A pointer to the field.

record_idp
The element ID, in case of array field is returned if not **NULL**.

cookiep
The cookie for the next field.

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

Return Values

Returns a pointer to the value on the flist. The pointer must be cast appropriately depending on the type of the field. Returns **NULL** if an error occurred or if the field is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_ADD

This macro adds a specified array element to the flist. The flist for the element fields is created and returned. The pointer to this element flist can then be used to set/put fields into the element.

If the specified array element already exists on the flist, the existing element flist is destroyed and replaced by the new element flist.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_ELEM_ADD(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                v_int32          elem_id,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist receiving the array element.

fld
The number of the field being added.

elem_id
The element ID of the element being added.

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

Return Values

Returns a pointer to the flist for the array element. Returns **NULL** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_ELEM_COPY

This macro copies an element in an array from one flist to another. You can change the element name and record ID while copying the element. The type must remain the same.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_ELEM_COPY(
                pin_flist_t    *src_flistp,
                pin_fld_num_t   src_fld,
                pin_rec_id_t    src_recID,
                pin_flist_t    *dest_flistp,
                pin_fld_num_t   dest_fld,
                pin_rec_id_t    dest_recID,
                pin_errbuf_t    *ebufp );
```

Parameters

src_flistp
A pointer to the source flist from which the element is copied.

src_fld
The element that is copied from the source flist.

src_recID
The record ID of the element that is copied.

dest_flistp
A pointer to the destination flist to which an element is copied.

dest_fld
The copied element in the destination flist.

dest_recID
The record ID of the copied element in the destination flist.

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

Return Values

Returns **1** if the field to be copied is found. Returns **0** if the field to be copied is not found. Not finding a field does not result in an error buffer error.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_COUNT

This macro counts the number of elements of an array on an flist. It does not look at substructure flists, so the elements must be on the flist passed in at the highest level.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_ELEM_COUNT(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist being counted.

fld
The field number of the array containing the elements being counted. Each time a field with this number is found, the element count is incremented.

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

Return Values

Returns the number of elements found as an unsigned integer. Returns **0** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_DROP

This macro drops the specified array element from an flist. The element flist is destroyed and the memory reallocated.

Important: This opcode causes an array to shift its indexing if an element other than the last is dropped. Do not use this PIN_FLIST_ELEM_DROP in a loop of PIN_FLIST_ELEM_GET_NEXT calls; the off-set will cause elements to be skipped.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_ELEM_DROP(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            elem_id,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the array element being removed.

fld
The field number of the array containing the element being removed.

elem_id
The element ID of the element being removed.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_GET

This macro gets the value of a specific array element from the flist. The element remains on the flist unchanged, and the value returned is a pointer to the element flist owned by the flist. The element flist returned *must* be treated as read-only to maintain the integrity of the flist. If a writable copy of the element flist is needed, the application must either make a copy of the returned element flist or use PIN_FLIST_ELEM_TAKE to take ownership of the element from the flist.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_ELEM_GET(
                pin_flist_t      *flistp,
                pin_fld_num_t     fld,
                int32             elem_id,
                int32             optional,
                pin_errbuf_t     *ebufp);
```

Parameters

flistp
A pointer to the flist containing the array element being obtained.

fld
The field number of the array containing the element being obtained.

elem_id
The ID of the array you need returned.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set, and the element is not found, an error condition is set.

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

Return Values

Returns a pointer to the element flist. Returns **NULL** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_GET_NEXT

This macro gets an array element from an flist. That is, this macro gets the value of the *next* element of a specified array on an flist. Lets the application walk the flist, retrieving each element of an array without knowing the element IDs ahead of time.

The element remains on the flist unchanged, and the value returned is a pointer to the element flist owned by the flist. The element flist returned *must* be treated as read-only to maintain the integrity of the flist. If a writable copy of the element flist is needed, the application must either make a copy of the returned element flist or use PIN_FLIST_ELEM_TAKE_NEXT to take ownership of the element from the flist.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_ELEM_GET_NEXT(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32           *elem_idp,
                int32            optional,
                pin_cookie_t    *cookie,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the array element being obtained.

fld
The field number of the array containing the element being taken.

elem_idp
A pointer to the number of the array element being taken.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

cookie
If set to **NULL**, the first element on the list is returned. Subsequent calls to this macro pass in the cookie, and the next element of the array is retrieved.

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

Return Values

Returns a pointer to the element flist, *elem_idp*, as the element number. Returns **NULL** if an error occurred or if the element is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_MOVE

This macro moves an element of an array from one flist to another. You can change the field name and record ID when you move the element. The type must remain the same.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_ELEM_MOVE(
                pin_flist_t     *src_flistp,
                pin_fld_num_t    src_fld,
                pin_rec_id_t     src_recID,
                pin_flist_t     *dest_flistp,
                pin_fld_num_t    dest_fld,
                pin_rec_id_t     dest_recID,
                pin_errbuf_t    *ebufp );
```

Parameters

src_flistp
A pointer to the source flist from which the element is moved.

src_fld
The element that is moved from the source flist.

src_recID
The record ID of the element that is moved.

dest_flistp
A pointer to the destination flist to which an element is moved.

dest_fld
The moved element in the destination flist.

dest_recID
The record ID of the moved element in the destination flist.

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

Return Values

Returns **1** if the field to be moved is found. Returns **0** if the field to be moved is not found. Not finding a field does not result in an error buffer error.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_PUT

This macro puts an array element on an flist. The element flist provided is used as the value of the array element. Ownership of the element flist is passed to the target flist, so the application must not destroy it once it has been put. The memory holding the value must be dynamically allocated.

After the value of the field has been added to an flist using this macro, the caller can no longer access the value directly using the pointer to the value. The flist management system may optimize memory usage by moving where the value is stored, so the original pointer is no longer valid.

If the specified array element already exists on the flist, the existing element flist is destroyed and replaced by the new element flist.

If an error condition exists or this macro otherwise fails, the element being put is destroyed. The memory is deallocated and an error is returned to the error buffer.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_ELEM_PUT(
                pin_flist_t     *flistp,
                pin_flist_t     *elem_flistp,
                pin_fld_num_t    fld,
                int32            elem_id,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the destination flist.

elem_flistp
A pointer to the flist containing the array element being added.

fld
The field number of the array receiving the element.

elem_id
The number of the element being put on the flist.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_SET

This macro sets a copy of an element on an flist. A dynamic copy of the specified element is made for the flist. The element passed in does not have to be in dynamic memory. The element passed in is unaffected by this macro. If the specified element already exists on the flist, the existing element is destroyed and replaced by the new element.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_ELEM_SET(
                pin_flist_t     *flistp,
                void            *elem_flistp,
                pin_fld_num_t    fld,
                int32            elem_id,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the destination flist for the element.

elem_flistp
A pointer to the flist for the input element.

fld
The field number of the array receiving the element.

elem_id
The number of the element being added.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_TAKE

This macro takes the value of an array element from an flist and removes it from the flist. The dynamically allocated memory holding the element flist is returned to the application. The application is then responsible for freeing this element flist when it is no longer needed. This macro is useful when the array element is no longer needed on the flist after the value is retrieved.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_ELEM_TAKE(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            elem_id,
                int32            optional,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the element being taken.

fld
The field number of the array whose element is being taken.

elem_id
The number of the element being taken.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

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

Return Values

Returns a pointer to the element flist. Returns **NULL** if an error occurred or the element is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_ELEM_TAKE_NEXT

This macro takes the value of the *next* element of an array from the flist. Lets the application walk the flist, retrieving each element of an array without knowing the element IDs ahead of time.

The element is removed from the flist. The dynamically allocated memory holding the element flist is returned to the application. The application is then responsible for freeing this element flist when it is no longer needed by the application. This macro is useful when the array element will not be needed on the flist after the value is retrieved.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_ELEM_TAKE_NEXT(
                pin_flist_t      *flistp,
                pin_fld_num_t     fld,
                int32            *elem_idp,
                int32             optional,
                pin_cookie_t     *cookie,
                pin_errbuf_t     *ebufp);
```

Parameters

flistp
A pointer to the flist of the array containing the element being taken.

fld
The field number of the array containing the element being taken.

elem_idp
A pointer to the number of the element being taken.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

cookie
If set to **NULL**, the first element on the list is returned. Subsequent calls to this macro pass in the cookie, and the next element of the array is retrieved.

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

Return Values

Returns a pointer to the element flist, *elem_idp*, as the element number. Returns **NULL** if an error occurred or if the element is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_FLD_COPY

This macro copies a field from one flist to another. If this macro is called to copy an array, it copies the array with all the elements in the array.

You can change the field name while copying the field. The type must remain the same.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_FLD_COPY(
                pin_flist_t     *src_flistp,
                pin_fld_num_t    src_fld,
                pin_flist_t     *dest_flistp,
                pin_fld_num_t    dest_fld,
                pin_errbuf_t    *ebufp);
```

Parameters

src_flistp
A pointer to the source flist from which the field is copied.

src_fld
The field that is copied from the source flist.

dest_flistp
A pointer to the destination flist to which a field is copied.

dest_fld
The copied field in the destination flist.

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

Return Values

Returns **1** if the field to be moved is found. Returns **0** if the field to be moved is not found. Not finding a field does not result in an error buffer error.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_FLD_DROP

This macro removes a field from an flist, destroying the value of the field and reallocating the memory.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_FLD_DROP(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the substructure.

fld
The field number of the substructure being removed.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_FLD_GET

This macro gets the value of a field from an flist. The value returned is a pointer to the actual value owned by the flist, and the field remains on the original flist, unchanged. The value returned must be treated as read-only to maintain the integrity of the flist. If a writable copy of the value is needed, the application must either make a copy of the returned value or use PIN_FLIST_FLD_TAKE to take ownership of the field from the flist.

Caution: The pointer returned is valid only until you modify the flist by setting a field, retrieving a field, or destroying the flist. To ensure that you have a valid pointer, always use PIN_FLIST_FLD_GET immediately before you use the field, or dereference the pointer returned from PIN_FLIST_FLD_GET and store the value locally.

Important: To copy a field from one flist to another, use PIN_FLIST_FLD_COPY instead of PIN_FLIST_FLD_GET and PIN_FLIST_FLD_SET. To copy an element from one flist to another, use PIN_FLIST_ELEM_COPY.

Syntax

```
#include "pcm.h"
void *
PIN_FLIST_FLD_GET(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            optional,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the field being obtained.

fld
The number of the field being obtained.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

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

Return Values

Returns a pointer to the value on the flist. The pointer must be cast appropriately depending on the type of the field. Returns **NULL** if an error occurred or if the field is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_FLD_MOVE

This macro moves a field from one flist to another. If this macro is called to move an array, it moves the array with all the elements in the array.

You can change the field name while moving the field. The type must remain the same.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_FLD_MOVE(
                pin_flist_t     *src_flistp,
                pin_fld_num_t    src_fld,
                pin_flist_t     *dest_flistp,
                pin_fld_num_t    dest_fld,
                pin_errbuf_t    *ebufp );
```

Parameters

src_flistp
A pointer to the source flist from which a field is moved.

src_fld
The field that is moved from the source flist.

dest_flistp
A pointer to the destination flist into which a field is moved.

dest_fld
The moved field in the destination flist.

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

Return Values

Returns **1** if the field to be moved is found. Returns **0** if the field to be moved is not found. Not finding a field does not result in an error buffer error.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_FLD_PUT

This macro puts a field (including its data value) in an flist. The memory holding the value must be dynamically allocated. The dynamic memory holding the value is given to the flist as part of the put. This is useful for adding a field to the flist without copying its value, if that memory is no longer needed by the application.

> **Important:** To move fields between flists or to rename fields, use PIN_FLIST_FLD_MOVE, PIN_FLIST_ELEM_MOVE, and PIN_FLIST_FLD_RENAME instead of PIN_FLIST_FLD_TAKE and PIN_FLIST_FLD_PUT.

After the value of the field has been added to an flist using this macro, the caller can no longer access the value directly using the pointer to the value. The flist management system may optimize memory usage by moving where the value is stored, so the original pointer is no longer valid.

If the specified field already exists in the flist, the previous value is destroyed and replaced by the new value.

If an error condition exists or this macro otherwise fails, the field being put is destroyed. The memory is deallocated and an error is returned to the error buffer.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_FLD_PUT(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                void            *valp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist receiving the field.

fld
The number of the field being added.

valp
A pointer to the field value being added.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_FLD_RENAME

This macro changes the name of a field in an flist. If you are changing the name of an array, this macro changes the names of all the elements in the array.

The type of the fields must be the same.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_FLD_RENAME(
                pin_flist_t     *flistp,
                pin_fld_num_t    src_fld,
                pin_fld_num_t    dest_fld,
                pin_errbuf_t    *ebufp)
```

Parameters

flistp
A pointer to the flist in which a field is renamed.

src_fld
The field that is renamed.

dest_fld
The new name of the field.

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

Return Values

This macro returns nothing.

Error Handling

If the field is not found, the error buffer contains a PIN_ERR_NOT_FOUND error.

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_FLD_SET

This macro adds a field and a value to an flist. A dynamic copy of the specified value is made for the flist. The value passed does not have to be in dynamic memory. The value passed is unaffected by the macro.

If the specified field already exists in the flist, the existing value is destroyed and replaced by the new value.

Important: To copy a field from one flist to another, use PIN_FLIST_FLD_COPY instead of PIN_FLIST_FLD_GET and PIN_FLIST_FLD_SET. To copy an element from one flist to another, use PIN_FLIST_ELEM_COPY.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_FLD_SET(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                void            *valp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist receiving the field.

fld
The number of the field being added.

valp
A pointer to the field value.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_FLD_TAKE

This macro takes a field from an flist and returns its value. The dynamically allocated memory holding the field value is returned to the application. The application is then responsible for freeing this memory when it is no longer needed. This macro is useful when fields will not be needed after the field value is retrieved.

> **Caution:** If you use PIN_FLIST_FLD_GET, you should do so before using this macro. PIN_FLD_FLIST_TAKE can modify the memory locations of the flist, making the PIN_FLIST_FLD_GET pointer invalid. To ensure that the pointer to the flist remains valid, always call PIN_FLIST_FLD_GET immediately before using the field.

Use PIN_FLIST_FLD_GET when a read-only pointer to the field is needed.

> **Important:** To move fields between flists or to rename fields, use PIN_FLIST_FLD_MOVE, PIN_FLIST_ELEM_MOVE, and PIN_FLIST_FLD_RENAME instead of PIN_FLIST_FLD_TAKE and PIN_FLIST_FLD_PUT.

Syntax

```
#include "pcm.h"
void *
PIN_FLIST_FLD_TAKE(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            optional,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the field being taken.

fld
The number of the field being taken.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

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

Return Values

Returns a pointer to the field's value. The pointer must be cast appropriately depending on the type of field. Returns **NULL** if an error occurred or if the field is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_ADD

This macro adds a substructure to an flist. The flist for the substructure is created and returned. The pointer to this substruct flist can then be used to set/put fields into the substructure. If the substructure already exists on the flist, the existing substruct flist is destroyed and replaced by the new substruct flist.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_SUBSTR_ADD(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist receiving the substructure.

fld
The field number of the substructure being added.

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

Return Values

Returns a pointer to the flist for the substructure. Returns **NULL** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_DROP

This macro removes a substructure from an flist, freeing the allocated memory.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_SUBSTR_DROP(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the substructure being dropped.

fld
The field number of the substructure being dropped.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_GET

This macro gets a substructure from an flist. The substructure remains on the flist unchanged, and the value returned is a pointer to the substructure flist, owned by the flist. The substructure returned *must* be treated as read-only to maintain the integrity of the flist. If a writable copy of the substructure flist is needed, the application must either make a copy of the returned substructure flist or use the PIN_FLIST_SUBSTR_TAKE macro to take ownership of the substructure.

Syntax

```
#include "pcm.h"
void *
PIN_FLIST_SUBSTR_GET(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            optional,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist with the substructure being obtained.

fld
The field number of the substructure being obtained.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

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

Return Values

Returns a pointer to the substructure flist. Returns **NULL** if an error occurred or if the element is not found.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_PUT

This macro puts a substructure on an flist. The substructure flist provided is used as the value of the substructure. Ownership of the substructure flist is passed to the target flist, so the application must not destroy it once it has been put. The memory holding the value must be dynamically allocated.

After the value of the field has been added to an flist using this macro, the caller can no longer access the value directly using the pointer to the value. The flist management system may optimize memory usage by moving where the value is stored, so the original pointer is no longer valid.

If the specified substructure already exists on the target flist, the existing element is destroyed and replaced by the new element.

If an error condition exists or the macro otherwise fails, the substructure being put is destroyed. The memory is deallocated and an error is returned to the error buffer.

This macro is optimal for adding inordinately large chunks of data to an flist. The flist does not allocate memory for the added data; it is merely linked to where the memory is already dynamically allocated. In contrast, PIN_FLIST_SUBSTR_SET adds an element by reallocating memory for it in the flist.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_SUBSTR_PUT(
                pin_flist_t     *flistp,
                void            *substr_flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist being added.

substr_flistp
A pointer to the flist containing the substructure being added.

fld
The field number of the substructure being added.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_SET

This macro adds a copy of a substructure to an flist. A dynamic copy of the specified substructure is made for the flist. The substructure passed in does not have to be in dynamic memory. The substructure passed in is unaffected by this macro. If the specified field already exists on the flist, the existing substructure is destroyed and replaced by the new substructure.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_SUBSTR_SET(
                pin_flist_t     *flistp,
                void            *substr_flistp,
                pin_fld_num_t    fld,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist receiving the substructure.

substr_flistp
A pointer to the flist containing the substructure being added.

fld
The field number of the substructure being added.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SUBSTR_TAKE

This macro takes a substructure off of an flist and returns its value. The dynamically allocated memory holding the field value is returned to the application. The application is then responsible for freeing this memory when it is no longer needed. This macro is useful when fields will not be needed after the field value is retrieved.

Syntax

```
#include "pcm.h"
void *
PIN_FLIST_SUBSTR_TAKE(
                pin_flist_t     *flistp,
                pin_fld_num_t    fld,
                int32            optional,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist containing the substructure being taken.

fld
The field number of the substructure being removed from *flistp*.

optional
If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Flist Management Macros

This section describes flist management macros.

PIN_FLIST_CONCAT

This BRM macro appends a (source) flist to the end of another (destination) flist. No comparisons between the flists are performed, and the source flist remains unchanged.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_CONCAT(
                pin_flist_t     *dest_flistp,
                pin_flist_t     *src_flistp,
                pin_errbuf_t    *ebufp);
```

Parameters

dest_flistp
A pointer to the destination flist.

src_flistp
A pointer to the source flist.

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

Return Values

Returns the concatenated flist in *dest_flistp*. If *src_flistp* is **NULL**, *dest_flistp* is returned unchanged. Returns an error in the error buffer if *dest_flistp* is **NULL**.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_COPY

This macro copies all levels of an existing flist, including its array elements and substructures. The copied fields and their values are duplicated so no memory is shared between the two flists.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_COPY(
                pin_flist_t     *flistp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist to be copied.

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

Return Values

Returns a pointer to the new flist. Returns **NULL** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_COUNT

This macro counts the number of fields on the flist. Only fields on the main flist are included. Each array element and substruct is counted as a single element.

If PIN_FLIST_COUNT is called with the pointer to an array element or substruct, the number of fields at that level of the flist are counted.

Syntax

```
#include "pcm.h"
int32
PIN_FLIST_COUNT(
                pin_flist_t     *flistp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to an flist to count the fields of.

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

Return Values

Returns the number of fields as an unsigned integer. Returns **0** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_CREATE

This BRM macro creates an flist that is used to pass parameters to the PCM_OP function. This macro creates an flist and returns a pointer that is used to reference the flist by all future operations. All memory for the flist is dynamically allocated.

Syntax

```
#include "pcm.h"
pin_flist_t *
PIN_FLIST_CREATE(ebufp)
                pin_errbuf_t     *ebufp);
```

Parameter

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

Return Values

Returns a pointer to the flist, in the form of **pin_flist_t***. Returns **NULL** if an error occurred.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_DESTROY

This macro destroys an flist. Flists use dynamically allocated memory, and they must be destroyed to free that memory. This macro destroys the entire contents of an flist, including all fields on the flist.

PIN_FLIST_DESTROY can destroy an flist, even if the error buffer is **NULL**.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_DESTROY(
                pin_flist_t     *flistp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist to destroy.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_DESTROY_EX

This macro destroys an flist. Flists use dynamically allocated memory, and they must be destroyed to free that memory. This macro first checks whether the pointer passed in is **NULL**. If the pointer is **NULL**, it returns. If the pointer is not **NULL**, it destroys the entire contents of the flist, including all fields on the flist, and sets the flist pointer to **NULL**.

> **Note:** PIN_FLIST_DESTROY_EX can destroy an flist, even if the error buffer is **NULL**.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_DESTROY_EX(
                pin_flist_t     **flistpp,
                pin_errbuf_t     *ebufp);
```

Parameters

****flistpp***
A pointer to the flist to destroy.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_PRINT

This macro prints, in ASCII format, an flist to a file. All levels of the flist, including the contents of array elements and substructures, are printed. This is useful for debugging applications that build or manipulate flists.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_PRINT(
                pin_flist_t     *flistp,
                FILE            *fi,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist to print.

fi
A pointer to a file to print a message to. If the value of this pointer is **NULL**, the message is printed to stdout.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Example

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

PIN_FLIST_SORT

This macro sorts flists and is normally used to sort array elements. Arrays sorted may also be the result of a search.

The flist to be sorted usually represents an array of search results returned from PCM_OP_SEARCH. The **sort_flistp** parameter is an flist that you construct with **sort_parameter**, called PIN_FLD_RESULTS. It would look like:

```
PIN_FLD_RESULTS
        field 1
        field 2
         .
         .
         .
```

Then use *sort_default* to compare nonexistent fields to existing fields. If all of the result elements have field values, **0** can be passed as the value of *sort_default*.

In cases where a result element has a field value, and it is being compared to another result element with the same field, but no value:

- A negative *sort_default* means that the result element with the missing field value is sorted *before* the other in the sorted list.
- A positive *sort_default* means the missing field occurs *after* the other.
- A *sort_default* of **0** means that they are considered equal and order is arbitrary on the sorted list.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_SORT(
                pin_flist_t     *flistp,
                pin_flist_t     *sort_listp,
                int32            sort_default,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist being sorted. The flist should normally consist of an array so that the sort is performed on elements of the array. Each element of the array may be a list of fields; it is those fields that get sorted. When you call this macro, pass the exact array (flist) you want sorted, not the entire array.

sort_listp
A list of fields in each element in *flistp* to use as sort fields. Elements in *flistp* are sorted in this order. If the value of this parameter is **NULL**, PIN_ERR_BAD_ARG is returned.

sort_default
The comparison to be used if an element is not found:

- f1 NOT found, f2 found - return *sort_default*
- f1 found, f2 NOT found - return *-sort_default*
- f1 NOT found, f2 NOT found - return 0 (equal)

- a negative value for *sort_default* means: f1 < f2
- a positive value for *sort_default* means: f1 > f2
- a zero value for *sort_default* means: f1 == f2

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_SORT_REVERSE

This macro sorts flists in reverse order. This macro, along with PIN_FLIST_SORT, is normally used to sort array elements. Arrays sorted may also be the result of a search.

The flist to be sorted usually represents an array of search results returned from PCM_OP_SEARCH or PCM_OP_STEP_SEARCH. The **sort_flistp** parameter is an flist that you construct with **sort_parameter**, called PIN_FLD_RESULTS. It would look like:

```
PIN_FLD_RESULTS
        field n
         .
         .
         .
        field 2
        field 1
```

Then use the *sort_default* parameter to compare nonexistent fields to existing fields. If all of the result elements have field values, **0** can be passed as the value of *sort_default*.

In cases where a result element has a field value, and it is being compared to another result element with the same field, but no value:

- A negative *sort_default* means that the result element with the missing field value is sorted *after* the other in the sorted list.
- A positive *sort_default* means the missing field occurs *before* the other.
- A *sort_default* of **0** means that they are considered equal and order is arbitrary on the sorted list.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_SORT_REVERSE(
                pin_flist_t     *flistp,
                pin_flist_t     *sort_listp,
                int32            sort_default,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist being sorted. The flist should normally consist of an array so that the sort is performed on elements of the array. Each element of the array may be a list of fields; it is those fields that get sorted.

sort_listp
A list of fields in each element in *flistp* to use as sort fields. Elements in *flistp* are sorted in this order. If the value of this parameter is **NULL**, PIN_ERR_BAD_ARG is returned.

sort_default
The comparison to be used if an element is not found:

- a zero value for *sort_default* means: f1 == f2
- a positive value for *sort_default* means: f1 > f2
- a negative value for *sort_default* means: f1 < f2

- f1 NOT found, f2 NOT found - > return 0 (equal)
- f1 found, f2 NOT found -> return *-sort_default*
- f1 NOT found, f2 found -> return *sort_default*

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_STR_TO_FLIST

This macro takes a string representation of an flist (for example, the output of PIN_FLIST_TO_STR) and creates an flist run-time data structure.

Syntax

```
#include "pcm.h"
void
PIN_STR_TO_FLIST(
                char            *str,
                int64            default_db,
                pin_flist_t    **flistp,
                pin_errbuf_t    *ebufp);
```

Parameters

str
A pointer to a string containing an flist in ASCII form.

default_db
A specified database number. If the ASCII string contains the sub-string "$DB", the database number in this parameter will replace it.

flistp
A pointer to a buffer for the return flist.

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

Return Values

Returns the string in *flistp*.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_TO_STR

This macro prints, in ASCII format, the contents of an flist to a buffer.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_TO_STR(
                pin_flist_t     *flistp,
                char           **strpp,
                int32           *lenp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist to print to a string.

strpp
A pointer to a buffer for the return string. If the value is **NULL**, a buffer is allocated using malloc.

lenp
The length of the buffer that *strpp* points to. The buffer must be large enough to include a **\0**. If the value of *strpp* is **NULL**, *len* is passed back as the size of the allocated buffer, including the **\0**.

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

Return Values

Returns the string in *strpp*. If a buffer was allocated, *len* is the size of the string, including the **NULL** terminator. If a buffer is allocated, the application owns the memory and must free it eventually.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_TO_STR_COMPACT_BINARY

This macro prints, in compact binary form, the contents of an flist to a buffer.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_TO_STR_COMPACT_BINARY(
                pin_flist_t     *flistp,
                char           **strpp,
                int32           *lenp,
                pin_errbuf_t    *ebufp);
```

Parameters

flistp
A pointer to the flist to print to a string.

strpp
A pointer to a buffer for the return string. If the value is **NULL**, a buffer is allocated using malloc.

lenp
The length of the buffer that *strpp* points to. The buffer must be large enough to include a **\0**. If the value of *strpp* is **NULL**, *len* is passed back as the size of the allocated buffer, including the **\0**.

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

Return Values

Returns the string in *strpp*. The string is stored in binary format in compact form, which means the field numbers, instead of the field names, are stored in the buffer. If a buffer was allocated, *len* is the size of the string, including the **NULL** terminator. If a buffer is allocated, the application owns the memory and must free it eventually.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_FLIST_TO_XML

This macro converts an flist to XML format. It is designed for converting an invoice to an XML format. The formatted XML invoice is generated directly from the flist. It ignores and does not convert data in buffer fields or fields of type PIN_FLDT_BINSTR.

Note: This macro does not generate a **.DTD** file.

Syntax

```
#include "pcm.h"
void
PIN_FLIST_TO_XML(
                 pin_flist_t     *flistp,
                 int32            flags,
                 int32            encoding,
                 char           **bufpp,
                 int             *lenp,
                 char            *root_elemname,
                 pin_errbuf_t   *ebufp);
```

Parameters

flistp
A pointer to the flist to convert.

flags
Specifies the name-attribute pairs to use for the XML element tag:

- PIN_XML_BY_TYPE
- Uses the **TYPE** field for the name of the XML element tag. This is the default.
- PIN_XML_BY_NAME
- Uses the field name for the name of the XML element tag.
- PIN_XML_BY_SHORT_NAME
- Uses the field name for the name of the XML element tag and drops the common prefix to include only the unique portion. For example, PIN_FLD_NAME becomes NAME.
- PIN_XML_FLDNO
- Uses the field number for the attribute of the XML element tag.
- PIN_XML_TYPE

  Uses the **TYPE** field for the attribute of the XML element tag.

encoding
Specify **UTF8**.

bufpp
A pointer to the buffer that will contain the XML converted data.

lenp
The size of the buffer that *bufpp* points to.

root_elemname
The root element name. If you do not specify this field, the default root element name, **document**, is used.

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

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

POID Management Macros

This section describes POID management macros.

PIN_POID_COMPARE

This BRM macro compares two POIDs for equality. All fields of the POIDs, including the revision level, must be identical for them to be considered equal.

Syntax

```
#include "pcm.h"
int32
PIN_POID_COMPARE(
               poid_t           *poidp1,
               poid_t           *poidp2,
               int32             check_rev,
               pin_errbuf_t     *ebufp);
```

Parameters

poidp1
A pointer to the first POID to be compared.

poidp2
A pointer to the second POID to be compared.

check_rev
Determines whether or not the revision level of two POIDs is compared. If *check_rev* is set to **0**, only the POID ID, database number, and type are compared. If *check_rev* is set to a non-zero value, the POID ID, database number, type, and revision number are compared.

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

Return Values

Returns **0** if the POIDs are identical. Returns a negative value if *poidp1* is less than *poidp2*. Returns a positive value if *poidp1* is greater than *poidp2*.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_COPY

This macro copies a POID. The new POID uses dynamically allocated memory and is owned by the caller.

If *src_poidp* is **NULL**, or if the source POID data **type** is **NULL**, a **NULL** value is returned, and no error condition is set.

Syntax

```
#include "pcm.h"
poid_t*
PIN_POID_COPY(
               poid_t           *src_poidp,
               pin_errbuf_t     *ebufp);
```

Parameters

src_poidp
A pointer to the source POID.

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

Return Values

Returns a pointer to the newly created POID if the macro is successful. Returns **NULL** if the macro fails.

Success codes

PCM_ERR_NONE

Error codes

PCM_ERR_NO_MEM

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_CREATE

This macro creates a POID. The POID uses dynamically allocated memory, and ownership of the POID is given to the caller. A copy is made of *type*, so it does not need to be in dynamic memory when passed.

id is typically initialized as **0**. The create operation finds the next available ID in the database and uses it when creating the object.

A source POID with a *type* of **NULL** is handled correctly. See the "Portal Object ID (POID)" in *BRM Developer's Guide* for more information on POIDs.

Syntax

```
#include "pcm.h"
poid_t*
PIN_POID_CREATE(
                int64             db,
                char             *type,
                int64             id,
                pin_errbuf_t     *ebufp);
```

Parameters

db
The database number.

type
The data type for the new POID. See the list of objects in "Storable Class Definitions". Examples are **/service** and **/event/customer/nameinfo**.

id
A unique object ID. This is a 64-bit quantity, so an extremely large number of objects can exist within a single database. Object IDs are unique within a single database, but not across databases.

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

Return Values

Returns a pointer to the newly created POID if the macro is successful. Returns **NULL** if the macro fails.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Examples

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

PIN_POID_DESTROY

This macro destroys a POID. POIDs use dynamically allocated memory and must be destroyed to free that memory. The entire POID is destroyed, including the **type** string.

Syntax

```
#include "pcm.h"
void
PIN_POID_DESTROY(
               poid_t           *poidp,
               pin_errbuf_t     *ebufp);
```

Parameters

poidp
A pointer to the POID to be destroyed.

ebufp
A pointer to an error buffer. Used to pass status information back to the caller. This parameter is optional. If a **NULL** is passed in, no error information is returned.

Return Values

This macro returns nothing.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

Examples

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

PIN_POID_FROM_STR

This macro converts a string to a POID.

> **Note:** This macro allocates the new POID's memory. To avoid memory leaks, PUT the POID onto an flist (typical case) or destroy the flist.

Syntax

```
#include "pcm.h"
poid_t*
PIN_POID_FROM_STR(
               char             *strp,
               char            **endcpp,
               pin_errbuf_t     *ebufp);
```

Parameters

strp
A pointer to the destination string.

endcpp
A pointer to the character following the last character of the POID value. That is, the character that terminated the scan (usually **NULL**, white space, or a new line).

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

Return Values

Returns a pointer to the POID created from the input string if the macro is successful. Returns **NULL** if the macro fails.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_GET_DB

This macro returns the database number portion of a POID.

Syntax

```
#include "pcm.h"
int64
PIN_POID_GET_DB(
               poid_t     *poidp);
```

Parameter

poidp
A pointer to the POID whose database number is being returned.

Return Values

Returns the database number if the macro is successful.

Error Handling

This macro does not handle errors.

PIN_POID_GET_ID

This macro returns a POID's ID.

Syntax

```
#include "pcm.h"
int64
PIN_POID_GET_ID(
                poid_t     *poidp);
```

Parameter

poidp
A pointer to the POID whose ID is being returned.

Return Values

Returns the POID's ID if the macro is successful.

Error Handling

This macro does not handle errors.

PIN_POID_GET_REV

This macro returns the POID's revision level. The revision level is incremented each time any portion of the object is updated.

Syntax

```
#include "pcm.h"
int32
PIN_POID_GET_REV(
               poid_t     *poidp);
```

Parameter

poidp
A pointer to the POID whose non-zero revision level is being returned.

Return Values

Returns the POID's revision level if the macro is successful.

Error Handling

This macro does not handle errors.

PIN_POID_GET_TYPE

This macro returns the object type of the POID in string format. Possible types are listed in "Storable Class Definitions". Examples are /**account** and **/event/billing/charge**.

Syntax

```
#include "pcm.h"
char*
PIN_POID_GET_TYPE(
               poid_t     *poidp);
```

Parameter

poidp
A pointer to the POID whose type is being returned.

Return Values

Returns the POID's type as a string if the macro is successful.

Error Handling

This macro does not handle errors.

PIN_POID_IS_NULL

This macro checks a POID to see whether it is **NULL**. The condition is satisfied if the pointer is **NULL** or the database number is **0**.

Syntax

```
#include "pcm.h"
int32
PIN_POID_IS_NULL(
               poid_t     *poidp);
```

Parameter

poidp
A pointer to the POID to check.

Return Values

Returns a non-zero value if the POID pointer is **NULL** or the database number is **0**.

Error Handling

This macro does not handle errors.

PIN_POID_LIST_ADD_POID

This macro adds a POID to the POID list.

Syntax

```
#include "pcm.h"
void
PIN_POID_LIST_ADD_POID(
                        char            **strpp,
                        poid_t          *pdp,
                        int32           flag,
                         pin_errbuf_t    *ebufp)
```

Parameters

strpp
Pointer to the POID list.

pdp
Pointer to the POID to be added to the list.

flag
A PCM flag (PCM_FLDFLG_FIFO or PCM_FLDFLG_CMPREV).

ebufp
Pointer to the error buffer.

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_COPY

This macro copies a POID list.

Syntax

```
#include "pcm.h"
poid_list_t *
PIN_POID_LIST_COPY(
                   poid_list_t  *src_pldp,
                    pin_errbuf_t  *ebufp)
```

Parameters

src_pldp
Pointer to the POID list to be copied.

ebuf
Pointer to the error buffer.

Return Values

Returns a pointer to the newly created POID list if the macro is successful. Returns NULL if the macro fails.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_COPY_NEXT_POID

This macro copies 'next' POID from the POID list.

Syntax

```
#include "pcm.h"
poid_t *
pin_poid_list_get_next(
                       char            *strp,
                       int32           optional,
                       pin_cookie_t    *cookiep,
                        pin_errbuf_t    *ebufp)
```

Parameters

strp
Pointer to the POID list from which the next POID is to be copied.

optional
If this flag is set to a non-zero value and the element is not found, no error condition is set. If this flag is not set, and the element is not found, an error condition is set.

cookiep
The cookie for the next POID.

ebufp
Pointer to the error buffer.

Return Values

Returns a pointer to the newly created POID if the macro is successful. Returns NULL if the macro fails.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_COPY_POID

This macro copies the specified POID from the POID list.

Syntax

```
#include "pcm.h"
poid_t*
PIN_POID_LIST_COPY_POID(
                        char            *strp,
                        void            *vp,
                        int32           flags,
                         pin_errbuf_t    *ebufp)
```

Parameters

strpp
Pointer to the POID list.

vp
Pointer to the POID to be copied.

flags
A PCM flag (PCM_FLDFLG_CMPREV or PCM_FLDFLG_TYPE_ONLY) to check for the existence of the POID to be copied.

Ebufp
Pointer to the error buffer.

Return Values

Returns a pointer to the newly created POID if the macro is successful. Returns NULL if the macro fails.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_CREATE

This macro creates a POID list.

Syntax

```
#include "pcm.h"
poid_list_t *
PIN_POID_LIST_CREATE(
                      pin_errbuf_t *ebufp)
```

Parameter

ebufp
Pointer to the error buffer.

Return Values

Returns a pointer to the newly created POID list if macro is successful. Returns NULL if the macro fails.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_DESTROY

This macro frees a POID list.

Syntax

```
#include "pcm.h"
void
PIN_POID_LIST_DESTROY(
                      poid_list_t *pldp,
                        pin_errbuf_t *ebufp)
```

Parameters

pldp
Pointer to the POID list to be freed.

ebufp
Pointer to the error buffer.

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_REMOVE_POID

This macro removes a POID from the POID list.

Syntax

```
#include "pcm.h"
void
PIN_POID_LIST_REMOVE_POID(
                          char        **strpp,
                          poid_t      *pdp,
                          int32       check_rev,
                            pin_errbuf_t *ebufp)
```

Parameters

strpp
Pointer to the POID list.

pdp
Pointer to the POID to be removed from the list.

check_rev
Determines the existence of the POID to be removed. If *check_rev* is set to 0, existence of the POID is checked.

ebufp
Pointer to the error buffer.

Return Values

This macro returns nothing.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_LIST_TAKE_NEXT_POID

This macro takes the 'next' POID from the POID list.

Syntax

```
#include "pcm.h"
poid_t *
pin_poid_list_take_next(
                        char            **strpp,
                        int32           optional,
                           pin_errbuf_t    *ebufp)
```

Parameters

strpp
Pointer to the POID list.

optional
If this flag is set to a non-zero value and the element is not found, no error condition is set. If this flag is not set, and the element is not found, an error condition is set.

ebufp
Pointer to the error buffer.

Return Values

Returns a pointer to the POID taken from the POID list if the macro is successful. Returns NULL if the macro fails.

Error Handling

This macro uses series-style ebuf error handling. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_PRINT

This macro prints a POID.

Syntax

```
#include "pcm.h"
void
PIN_POID_PRINT(
               poid_t           *poidp,
               FILE             *fi,
               pin_errbuf_t     *ebufp);
```

Parameters

poidp
A pointer to the POID to print.

fi
The **FILE** pointer to the file to receive the message. If the value of **FILE** is **NULL**, the message is printed to **stdout**.

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

Return Values

This macro returns nothing.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

PIN_POID_TO_STR

This macro prints a POID to a string. Put the info of a POID into a string (*strpp*). If the buffer (*ebufp*) is not large enough to hold the string, **PIN_ERR_BAD_ARG** is returned. The return value of *lenp* includes the **\0**. The format of the string is:

"%d %s %d %d"

where the values are for:

database_number object_type object_id object_revision_level

object_revision_level is incremented each time the object is updated.

Syntax

```
#include "pcm.h"
void
PIN_POID_TO_STR(
               poid_t           *poidp,
               char            **strpp,
               int32            *lenp,
               pin_errbuf_t     *ebufp);
```

Parameters

poidp
A pointer to the POID to be printed.

strpp
A pointer to the buffer receiving the string version of the POID. This should be 48 larger that the value of PCM_MAX_POID_TYPE, to accommodate the largest strings.

lenp
The length of the buffer.

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

Return Values

This macro returns nothing.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series ebuf–style API routines using the same error buffer, and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors. See "Understanding API Error Handling and Logging" in *BRM Developer's Guide* for details on error handling algorithms.

String Manipulation Functions

This section describes string manipulation functions.

About the String Manipulation Functions

You use the string manipulation functions to store and retrieve server strings, such as reason codes, help messages, and other text displayed in the user interface. These strings are stored on the server so that they can be easily localized for multiple languages and displayed simultaneously in the appropriate languages for the client locales. For example, French and German customer service representatives (CSRs) logged into BRM at the same time can read messages in their own languages.

String manipulation functions also allow data received by the database to be canonicalized for easy processing.

BRM Locale IDs

UNIX, Windows, and Java use different locale IDs. So BRM includes a locale table, which maps the BRM locale to locale strings for various platforms.

Similar to UNIX, the BRM locale is either:

- The two-character ISO code for the language. These two-character locales are used for a language in its country of origin. For example, **fr** designates French used in France.
- A concatenation of the two-character ISO code for the language and the two-character ISO code for the country. For example, **en_US** designates English in the United States.

The locale description IDs are mapped to a **/strings** table containing the textual description of the supported locales. This table and the BRM table name are stored in the database under **/config/locales**.

For more information on BRM locale names, see "Locale Names" in *BRM Developer's Guide*.

Storable Class Hierarchy for Localized Strings

BRM includes a **/strings** storable class to store localized strings.

Note: You cannot extend the **/strings** storable class.

Structure of the **/strings** storable class:

```
/strings
POID PIN_FLD_POID
TIMESTAMP PIN_FLD_CREATED_T
TIMESTAMP PIN_FLD_MOD_T
STRING PIN_FLD_DOMAIN             required, length = 1023
STRING PIN_FLD_DESCR              optional, length = 1023
STRING PIN_FLD_LOCALE             required, length = 1023
INT PIN_FLD_STRING_ID             required
INT PIN_FLD_STR_VERSION           required
STRING PIN_FLD_STRING             required, length = 1023
STRING PIN_FLD_HELP_STRING        optional, length = 1023
```

For descriptions of the fields, see the **/strings** storable class description.

Important: Do not change these names and numbers or the information will not be accessible.

Locale Mapping

For detailed information on BRM locale mapping, see "Locale Names" in *BRM Developer's Guide*.

Localized String Data Files

A file of localized string data contains multibyte character set (MBCS) strings, and the data is loaded into the database by running a utility that constructs storable string objects using information in the file.

The file extension of the file must be the BRM locale ID.

Sample names for files containing localized string data:

- **locale_descr.en_US** contains locale description information for United States English.
- **reasons.en_US** contains all of the reason code data for United States English.

String File Format Description

This section describes the required format of the string file. To use this file with the related functions and utilities, the file must follow this format.

Note: The load utility parser is case-insensitive to the keywords. It passes the locale and domain strings to the database as received. BRM is case sensitive. For example, **en_us** and the BRM locale **en_US** are not considered the same, nor are "Reason Codes-Credit Reasons" and "reason codes-credit reasons."

- Comments begin with the # symbol. All comments and white space are ignored.
- The string file has a locale ID as the first noncommented statement of the file, and there is only one locale ID per file. You can use existing domains in the files and/or add your own. Organize your strings by domains within the file.
- The string object definition is bounded by STR-END and consists of an ID unique within a domain, a string version, and the string itself.
- A string is delimited by quotation marks and can contain any character, including a quotation mark if escaped (\"). The percent symbol followed by an integer (%1) is interpreted as a substitution parameter flag.
- For reason codes, the version field specifies the domain of the reason, such as credit or debit.

This example shows a compatible string file:

```
#######################
# strings.en_US
#######################

LOCALE = "en_US" ;

DOMAIN = "Reason Codes-Credit Reasons" ;
```

```
STR
     ID = 1 ;
     VERSION = 1 ;
     STRING = "Customer not satisfied with service" ;
END
STR
     ID = 2 ;
     VERSION = 1 ;
     STRING = "Customer unaware of charges" ;
END
STR
     ID = 3 ;
     VERSION = 1 ;
     STRING = "Debited account by mistake" ;
END

DOMAIN = "Reason Codes-Debit Reasons" ;
STR
     ID = 1 ;
     VERSION = 1 ;
     STRING = "Technical and support charges" ;
END
STR
     ID = 2 ;
     VERSION = 1 ;
     STRING = "Service charges" ;
END
STR
     ID = 3 ;
     VERSION = 1 ;
     STRING = "Credited account by mistake" ;
END
```

String Manipulation Example

You can create message strings in multiple languages to obtain all the reason codes for English.

This is an example definition:

```
string_list_t*
pcm_get_localized_string_list(
               pcm_context_t     *context_p,
               const char        *locale_p,
               const char        *domain_p,
               const int32        string_id,
               const int32        string_vers,
               pin_errbuf_t      *ebufp);
```

The top-level function, **pcm_get_localized_string_list**, allows arbitrary queries on the **/strings** table. The argument list is similar to **pcm_get_localized_string** except that message buffers are not supplied by the caller. The function can accept a null locale string, a null domain string, a string ID = -1, or a string version = -1 to indicate that the argument is not part of the search.

This example shows retrieving strings:

```
pcm_get_localized_string_list(context_p,"en_US","Reason Codes-Active Status
Reasons",-1,1,ebufp);
```

is equivalent to:

```
select*
from strings_t
where locale = "en_US" AND
     domain = "Reason Codes-Active Status Reasons" AND
     string_vers = 1
```

which returns a set of string objects for any locale ID fitting these criteria. The function returns a container object of type **string_list_t**.

String Manipulation Functions

Table 2–5 lists String Manipulation Functions.

Table 2–5 String Manipulation Functions

Function	Description
pcm_get_localized_string_list	Retrieves the specified string list to be used by the string manipulation functions.
pin_string_list_destroy	Deallocates the object and its flist when finished with the string list.
pin_string_list_get_next	Retrieves the next object in the string list.

pcm_get_localized_string_list

This function retrieves the specified string list to be used by the string manipulation functions.

Use this function to obtain a group of related strings. It is much more efficient than calling **pcm_get_error_message** for each individual string.

pin_string_list_destroy

This function deallocates the object and its flist when finished with the string list.

> **Important:** To prevent memory leaks, you must call this after calling **pcm_get_string_list**.

Syntax

```
void
pin_string_list_destroy(
                string_list_t     *string_listp,
                pin_errbuf_t      *ebufp);
```

Parameters

string_listp
A pointer to the list.

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

pin_string_list_get_next

This function retrieves the next object in the string list.

The caller passes in the string list and a string info object, and the attributes of the next string object are pulled from the list and copied to the string info object. The info object is then returned to the caller. This function calls **pin_string_info_init** internally to flush the string info object and prepare it for new data. This allows the same string info object to be used repeatedly when iterating through the list.

Syntax

```
string_info_t*
pin_string_list_get_next(
               string_list_t     *string_listp,
               string_info_t     *string_infop,
               pin_errbuf_t      *ebufp);
```

Parameters

string_listp
A pointer to the list.

string_infop
A pointer to the string.

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

Validity Period Manipulation Macros

Validity period manipulation macros are used to get and set relative offset values for validity periods that start and end after a relative period passes. For example, a product's cycle fee period can become effective three months after the product is purchased.

About Relative Offset Values

Relative validity period information is stored in the BRM database in DETAILS fields. There are DETAILS fields for product, discount, and resource-balance validity periods. The specific name of the fields vary, but all end with "_DETAILS".

Relative validity period information includes the following values:

- Mode - Specifies generally when the validity period starts or ends and can be one of these:
  - PIN_VALIDITY_ABSOLUTE = 0
  - PIN_VALIDITY_IMMEDIATE = 1
  - PIN_VALIDITY_NEVER = 2
  - PIN_VALIDITY_FIRST_USAGE = 3
  - PIN_VALIDITY_RELATIVE = 4
- Unit - Specifies the type of offset unit, which can be one of these:
  - Seconds = 1
  - Minutes = 2
  - Hours = 3
  - Days = 4
  - Months = 5
  - Event cycles = 7
  - Accounting cycles = 8
  - Billing cycles = 9
  - None = 0
- Offset - Specifies the number of units in the offset period.

Note: Not all of the unit and mode values listed above can be used with every relative validity period in BRM. The unit and mode you can specify depends on the validity period you're setting and whether you're setting the start or end time. For more information, see the following topics:

- For information about the relative start and end times of products and discounts in price plans, see "Managing /deal Objects" in *BRM Setting Up Pricing and Rating*.
- For information about the relative start and end times of products and discounts owned by accounts, see "Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts" in *BRM Managing Customers*.
- For information about the relative start and end times of resource balances, see "Managing the Validity Period of Granted Resources" in *BRM Setting Up Pricing and Rating*.

PIN_VALIDITY_GET_UNIT

This macro retrieves the relative offset unit from the start- or end-time details value that is passed in.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_GET_UNIT(
                u_int32     encoded_value);
```

Parameter

encoded_value
The encoded value of the start- or end-time details field.

Return Values

Returns the value of the relative offset unit.

PIN_VALIDITY_GET_OFFSET

This macro retrieves the relative offset (the number of units in the relative period) from the start- or end-time details value that is passed in.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_GET_OFFSET(
                u_int32     encoded_value);
```

Parameter

encoded_value
The encoded value of the start- or end-time details field.

Return Values

Returns the value of the relative offset.

PIN_VALIDITY_GET_MODE

This macro retrieves the mode value from the start- or end-time details value that is passed in.

Syntax

```
#include "pcm.h"
pin_validity_modes_t
PIN_VALIDITY_GET_MODE(
                u_int32     encoded_value);
```

Parameter

encoded_value
The encoded value of the start- or end-time details field.

Return Values

Returns the value of the relative mode.

PIN_VALIDITY_SET_UNIT

This macro sets the relative offset unit in the start- or end-time details value that is passed in.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_SET_UNIT(
                 u_int32      encoded_value,
                  u_int32      unit_value);
```

Parameters

encoded_value
The encoded value of the start- or end-time details field.

unit_value
The offset unit value to set.

Return Values

Returns the encoded value of the start- or end-time details field set with the unit value passed in.

PIN_VALIDITY_SET_OFFSET

This macro sets the relative offset (number of offset units) in the start- or end-time details value that is passed in.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_SET_OFFSET(
                u_int32     encoded_value,
                 u_int32          offset_value);
```

Parameters

encoded_value
The encoded value of the start- or end-time details field.

offset_value
The offset value to set.

Return Values

Returns the encoded value of the start- or end-time details field set with the offset value passed in.

PIN_VALIDITY_SET_MODE

This macro sets the relative mode in the start- or end-time details value passed in.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_SET_MODE(
                u_int32     encoded_value,
                 pin_validity_modes_t     mode_value);
```

Parameters

encoded_value
The encoded value of the start- or end-time details field.

mode_value
The mode value to set.

Return Values

Returns the encoded value of the start- or end-time details field set with the mode value passed in.

PIN_VALIDITY_DECODE_FIELD

This macro decodes the values of the mode, unit, and offset in the start- or end-time details value passed in and then sets them in mode, unit, and offset variables.

Syntax

```
#include "pcm.h"
void
PIN_VALIDITY_DECODE_FIELD(
                u_int32                   encoded_value,
                pin_validity_modes_t    mode_variable,
                u_int32                   unit_variable,
                 u_int32                   offset_variable);
```

Parameters

encoded_value
The encoded value of the start- or end-time details field.

mode_variable
The mode variable to set.

unit_variable
The unit variable to set.

offset_variable
The offset variable to set.

Return Values

This macro returns nothing.

PIN_VALIDITY_ENCODE_FIELD

This macro takes the mode, unit, and offset values passed in and encodes them into a start- or end-time details field value.

Syntax

```
#include "pcm.h"
u_int32
PIN_VALIDITY_ENCODE_FIELD(
                pin_validity_modes_t    mode_value,
                u_int32                   unit_value,
                 u_int32                   offset_value);
```

Parameters

mode_value
The mode value.

unit_value
The unit value.

offset_value
The offset value.

Return Values

Returns the encoded value of the start- or end-time details field, set with the mode, unit, and offset values passed in.

3

Storable Class Definitions

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) storable class.

For more information about storable class definitions and field definitions, see *BRM Storable Class Reference*.

For information on how to define or modify storable classes and fields, see "Creating, Editing, and Deleting Fields and Storable Classes" in *BRM Developer's Guide*.

For related information, see "Storable Class-to-SQL Mapping" and "About Flists" in *BRM Developer's Guide*.

Fields Common to All Storable Classes

Every BRM storable class requires three fields to create its storable object in the system. These fields are available to BRM applications and Facilities Modules (FMs) but cannot be written to directly; they are manipulated only by the Storage Manager.

The fields are:

- PIN_FLD_POID. The unique ID for the object.
- PIN_FLD_CREATED_T. The time that the object was created.
- PIN_FLD_MOD_T. The last time the object was modified.

4

Perl Extensions to the PCM Libraries

This chapter contains a list of functions in **pcmif**, the Perl extension to Oracle Communications Billing and Revenue Management (BRM) Portal Communications Module (PCM) library, with links to the description of each function in the library.

For guidelines on using the Perl extensions to create applications, see "Creating Client Applications by Using Perl PCM" in *BRM Developer's Guide*.

For sample Perl scripts using **pcmif**, see "Example Perl Scripts".

Connection Functions

Table 4–1 list the connection function perl extensions to the PCM libraries.

Table 4–1 Connection Functions

Function	Description
pcm_context_close	Closes the given PCM context, disconnects from BRM, and frees memory associated with the context.
pcm_perl_connect	Connects to BRM by using PCM_CONNECT.
pcm_perl_context_open	Opens a PCM context to BRM by using PCM_CONTEXT_OPEN.
pcm_perl_get_session	Obtains the session ID set after login as a printable POID and returns it as a string.
pcm_perl_get_userid	Obtains the user ID set after login as a printable POID and returns it as a string.
pin_perl_time	Returns the time from the **pin_virtual_time** function, which is used to change time in BRM.

Error-Handling Functions

Table 4–2 list the error-handling function perl extensions to the PCM libraries.

Table 4–2 Error-Handling Functions

Function	Description
pcm_perl_destroy_ebuf	Deletes a previously created error buffer from memory.
pcm_perl_ebuf_to_str	Returns a static string with a printable representation of the error buffer.
pcm_perl_is_err	Checks for errors and returns the integer value of the error code in the error buffer.
pcm_perl_new_ebuf	Creates an empty error buffer structure and returns a pointer to it.
pcm_perl_print_ebuf	Executes a **printf** of the printable representation of the error buffer.
pin_set_err	Sets an error buffer.

Flist Conversion Functions

Table 4–3 list the flist conversion function perl extensions to the PCM libraries.

Table 4–3 Flist Conversion Functions

Function	Description
pin_flist_destroy	Deletes an opaque flist.
pin_flist_sort	Sorts the specified flist using PIN_FLIST_SORT.
pin_perl_flist_to_str	Converts an opaque flist into a printable string representation.
pin_perl_str_to_flist	Converts a printable flist into an opaque flist and returns a reference to the flist.

PCM Opcode Functions

Table 4–4 list the PCM opcode function perl extensions to the PCM libraries.

Table 4–4 PCM Opcode Functions

Function	Description
pcm_perl_op	Performs the indicated PCM operation with the given flags and input flist. It returns the resulting flist.

Example Perl Scripts

This section describes sample Perl scripts.

Perl Script Example 1

This sample script performs the following actions:

- It connects to BRM using the login information in the parameters set in the Config section. The **pin.conf** file only needs a dummy user ID entry.
- If there is an argument, it uses that as the POID ID of the data object to read.
- If there is no argument, it uses POID ID 1 as the default.
- It then reads an object with the POID ID using PCM_OP_READ_OBJ and displays the resulting flist.

```
#The first line of the Perl script.
#!/BRM_Home/perl/bin/perl
#
#Test a readobj of /data N (defaults to 1).
#Use the following two lines to specify the directory of the pcmif

#files and that you are using the pcmif module.

use lib '.' ;
use pcmif;

# Config section
# Uses pcm_context_open(), so requires pin.conf with userid only

# Set the login information.
 $LOGIN_DB = "0.0.0.1";
 $LOGIN_NAME = "root.0.0.0.1";
 $LOGIN_PASSWD = "password";
```

```
 $CM_HOST = "somehost";

# Setup and connect
# Create an ebuf for error reporting.

  $ebufp = pcmif::pcm_perl_new_ebuf();

# Use a "here" document to assign an flist string to a variable.

  $f1 = <<"XXX"
  0 PIN_FLD_POID POID [0] $LOGIN_DB /service/pcm_client 1 0
  0 PIN_FLD_TYPE               ENUM [0] 1
  0 PIN_FLD_LOGIN           STR [0] "$LOGIN_NAME"
  0 PIN_FLD_PASSWD_CLEAR    STR [0] "$LOGIN_PASSWD"
  0 PIN_FLD_CM_PTR     STR [0] "ip $CM_HOST 11960"
   XXX
   ;

# Use the string-to-flist conversion function to parse the flist string
# that contains the login information and use it to open a PCM #context.

$login_flistp = pcmif::pin_perl_str_to_flist($f1,
                                              $LOGIN_DB, $ebufp);

# Check for errors and print the error report.
          if (pcmif::pcm_perl_is_err($ebufp)) {
              print "flist conversion failed\n";
              pcmif::pcm_perl_print_ebuf($ebufp);
              exit(1);
              }
# Open a PCM context.
$pcm_ctxp = pcmif::pcm_perl_context_open($login_flistp,
                                              $db_no, $ebufp);
# Check for errors and print the status of the action.

  if (pcmif::pcm_perl_is_err($ebufp)) {
      pcmif::pcm_perl_print_ebuf($ebufp);
      exit(1);
     } else {
     $my_session = pcmif::pcm_perl_get_session($pcm_ctxp);
     $my_userid = pcmif::pcm_perl_get_userid($pcm_ctxp);
     print "back from pcmdd_context_open()\n";
     print "    DEFAULT db is: $db_no \n";
     print "    session poid is: ", $my_session, "\n";
     print "    userid poid is: ", $my_userid, "\n";
     }

# See if we should default to 1, or get a number

          if ($#ARGV >= 0) {
                $obj_id = $ARGV[0];
          } else {
                $obj_id = 1;
          }

# Build an flist.
          $f1 = <<"XXX"
          0 PIN_FLD_POID  POID [0] $db_no /data $obj_id 0
          XXX
          ;
```

```
# Convert the flist you built from a string to the flist format.

$flistp = pcmif::pin_perl_str_to_flist($f1, $db_no, $ebufp);

# Check for errors and print the error report.
          if (pcmif::pcm_perl_is_err($ebufp)) {
                print "flist conversion failed\n";
                pcmif::pcm_perl_print_ebuf($ebufp);
                exit(1);
          }
# Convert the flist to a printable string and print it.

          $out = pcmif::pin_perl_flist_to_str($flistp, $ebufp);
          print "IN flist is:\n";
          print $out;

# Perform a PCM operation to read an object and assign the result
# to a variable. Check for errors and print the error report.

$out_flistp = pcmif::pcm_perl_op($pcm_ctxp, "PCM_OP_READ_OBJ", 0,
              $flistp, $ebufp);
          if (pcmif::pcm_perl_is_err($ebufp)) {
                print "robj failed\n";
                pcmif::pcm_perl_print_ebuf($ebufp);
                exit(1);
          }
# Convert the flist for the object you read to a printable string and print it.

$out = pcmif::pin_perl_flist_to_str($out_flistp, $ebufp);
          print "OUT flist is:\n";
          print $out;

# Close the PCM context. Check for errors and print the error report.
          pcmif::pcm_context_close($pcm_ctxp, 0, $ebufp);
          if (pcmif::pcm_perl_is_err($ebufp)) {
                print "BAD close\n",
                    pcmif::pcm_perl_ebuf_to_str($ebufp), "\n";
                exit(1);
          }
          exit(0);
```

Perl Script Example 2

The following example is used to set up an account with a service of type **/service/ip** with the user name **testterm01** (for a test script). It checks for the existence of the service, and exits if the service is found. Otherwise, it finds the **/deal** needed for "IP Basic" (a standard default) and then creates the **/account** and **/service/ip** objects by using PCM_OP_CUST_COMMIT_CUSTOMER.

```
#!/BRM_Home/perl/bin/perl

# This is the directory for the pcmif.so and pcmif.pm files.
# For most usage this is not needed, since they will be obtained
# from the default directory (builtin to perl/BRM_Home/<vers>/lib).

use lib '.' ;

# The key - You MUST include this to indicate that you are using
# the pcmif extension.
```

```
use pcmif;

# The "pcmif::" prefix is a class prefix, meaning that the
# function "pcm_perl_new_ebuf()" is from the package/class
#"pcmif".
#
# Get an ebuf for error reporting.
#
$ebufp = pcmif::pcm_perl_new_ebuf();


# Do a pcm_connect(), $db_no is a return.

$pcm_ctxp = pcmif::pcm_perl_connect($db_no, $ebufp);


# Convert an ebuf to a printable string.

$ebp1 = pcmif::pcm_perl_ebuf_to_str($ebufp);

# Check for errors. Always do this.

if (pcmif::pcm_perl_is_err($ebufp)) {
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
} else {
print "back from pcm_connect()\n";
print "   DEFAULT db is: $db_no \n";
}

# NOTE: The following convention ($DB_NO) was established
# for use with testnap, to substitute the database number
# into a printed flist as it was parsed into testnap.
# We follow the text convention, but we let perl
# do the substitution via this variable (in upper case).
# NOTE: The flist parse should also perform
# this substitution since it gets fed $db_no.
# for testnap convention.
$DB_NO = $db_no;

# Use a "here" document to build an flist string into
# a variable. This flist will then be parsed and
# used in a pcm_op.
#
# search to see if /service/ip "testterm01" is already created

$f1 = <<"XXX"
0 PIN_FLD_POID           POID [0] $DB_NO /search 236 0
0 PIN_FLD_PARAMETERS     STR [0] "ip"
0 PIN_FLD_ARGS      ARRAY [1]
1     PIN_FLD_LOGIN        STR [0] "testterm01"
0 PIN_FLD_RESULTS      ARRAY [0]
1     PIN_FLD_POID    POID [0] 0.0.0.0  0 0
1     PIN_FLD_LOGIN        STR [0] ""
XXX
;
$flistp = pcmif::pin_perl_str_to_flist($f1, $db_no, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
print "flist conversion to check for testterm01 failed\n";
pcmif::pcm_perl_print_ebuf($ebufp);
```

```
exit(1);
}
$out_flistp = pcmif::pcm_perl_op($pcm_ctxp, "PCM_OP_SEARCH", 0, $flistp, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
print "SEARCH for testterm01 failed\n";
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
}

#
# Check if "testterm01" is there. If it is you do not
# have to recreate.
#
$out = pcmif::pin_perl_flist_to_str($out_flistp, $ebufp);
# XXX warning, no error check

pcmif::pin_flist_destroy($flistp);
pcmif::pin_flist_destroy($out_flistp);

# We converted the output flist into $out above,
# then cleaned the flist objects up.  Now we use
# a perl string matching operator to look for the
# user id we want.
#
if ($out =~ "testterm01") {
print "testterm01 already exists\n" ;
print $out;
exit(0);
}


print "XXX testterm01 does NOT exist\n" ;

#
# First we need the poid of the deal - use "IP Basic".
#
$f1 = <<"XXX"
0 PIN_FLD_POID           POID [0] $DB_NO /search 223 0
0 PIN_FLD_ARGS      ARRAY [1]
1     PIN_FLD_NAME        STR [0] "IP Basic"
0 PIN_FLD_RESULTS      ARRAY [0]
1     PIN_FLD_POID    POID [0] 0.0.0.0  0 0
XXX
;
#
$flistp = pcmif::pin_perl_str_to_flist($f1, $db_no, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
print "flist conversion to search for deal failed\n";
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
}
$out_flistp = pcmif::pcm_perl_op($pcm_ctxp, "PCM_OP_SEARCH", 0, $flistp, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
print "SEARCH for deal failed\n";
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
}

$out = pcmif::pin_perl_flist_to_str($out_flistp, $ebufp);
# XXX warning, no error check
```

```
pcmif::pin_flist_destroy($flistp);
pcmif::pin_flist_destroy($out_flistp);

if ($out !~ "/deal") {
print "no deal found \n" ;
print $out;
exit(1);
}

#
# The deal poid (which will be <db> /deal <id> <rev>)
# is isolated with index().Then the rest of the line
# (containing the id...) goes into deal_poid, which is
# trimmed by saving the matching pattern
# (ie the id number) and substituting the saved pattern
# (ie just the numbers) for the rest of the line.
#
$deal_at = index($out, "/deal");
$deal_poid = substr($out, $deal_at + 6);
$deal_poid =~ s|([0-9][0-9]*) .*|$1| ;

print "deal poid is ", $deal_poid, "\n";

#
# now we fill in an flist for COMMIT_CUSTOMER
#
$f1 = <<"XXX"
0 PIN_FLD_POIDPOID [0] $DB_NO /account 0
0 PIN_FLD_ACCOUNT_OBJPOID [0] $DB_NO /account 0
0 PIN_FLD_AAC_ACCESS STR [0] "setup.fm_term"
0 PIN_FLD_AAC_SOURCE STR [0] "setup.fm_term"
0 PIN_FLD_AAC_VENDOR STR [0] "setup.fm_term"
0 PIN_FLD_AAC_PACKAGE STR [0] "setup.fm_term"
0 PIN_FLD_AAC_PROMO_CODE STR [0] "setup.fm_term"
0 PIN_FLD_AAC_SERIAL_NUM STR [0] "setup.fm_term"
0 PIN_FLD_BILLINFOARRAY [1]
1 PIN_FLD_BILL_TYPEENUM [0] 0
1 PIN_FLD_CURRENCYUINT [0] 840
0 PIN_FLD_PAYINFOARRAY [1]
1 PIN_FLD_NAMEINFO_INDEXUINT [0] 1
0 PIN_FLD_NAMEINFOARRAY [1]
1 PIN_FLD_SALUTATION STR [0] "Mr."
1 PIN_FLD_LAST_NAME STR [0] "testterm01"
1 PIN_FLD_FIRST_NAME STR [0] "testterm01"
1 PIN_FLD_MIDDLE_NAME STR [0] "x"
1 PIN_FLD_TITLE STR [0] "title"
1 PIN_FLD_COMPANY STR [0] "company"
1 PIN_FLD_ADDRESS STR [0] "address"
1 PIN_FLD_CITY  STR [0] "Cupertino"
1 PIN_FLD_STATE         STR [0] "CA"
1 PIN_FLD_ZIP         STR [0] "95014"
1 PIN_FLD_COUNTRY STR [0] "USA"
1 PIN_FLD_EMAIL_ADDR STR [0] "email_addr"
1 PIN_FLD_CONTACT_TYPE STR [0] "contact_type"
0 PIN_FLD_SERVICESARRAY [1]
1 PIN_FLD_SERVICE_OBJPOID [0] $DB_NO /service/ip 0
1 PIN_FLD_LOGIN STR [0] "testterm01"
1 PIN_FLD_PASSWD_CLEAR STR [0] "testterm01"
```

```
XXX
;

#
# To avoid quotation problems in the above here document,
# the deal is appended via ".".
#
$f1 = $f1 . "1PIN_FLD_DEAL_OBJ POID [0] $DB_NO /deal $deal_poid" ;

print "flist is now\n";
print $f1;

$flistp = pcmif::pin_perl_str_to_flist($f1, $db_no, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
}
$out_flistp = pcmif::pcm_perl_op($pcm_ctxp, "PCM_OP_CUST_COMMIT_CUSTOMER",
0, $flistp, $ebufp);

if (pcmif::pcm_perl_is_err($ebufp)) {
print "BAD op: PCM_OP_CUST_COMMIT_CUSTOMER\n";
pcmif::pcm_perl_print_ebuf($ebufp);
exit(1);
}

$out = pcmif::pin_perl_flist_to_str($out_flistp, $ebufp);
print "OUT flist is \n" ;
print $out;

pcmif::pin_flist_destroy($flistp);
pcmif::pin_flist_destroy($out_flistp);

pcmif::pcm_context_close($pcm_ctxp, 0, $ebufp);
if (pcmif::pcm_perl_is_err($ebufp)) {
print "BAD close\n",
    pcmif::pcm_perl_ebuf_to_str($ebufp), "\n";
exit(1);
```

pcm_context_close

This function closes the given PCM context, disconnects from BRM, and frees memory associated with the context. If a context is no longer needed, make sure you close it.

For more information, see "PCM_CONTEXT_CLOSE".

Syntax

```
void
pcm_context_close(ctxp, how, ebufp);
```

Parameters

ctxp
A reference to an open PCM context.

how
Defines how to close the connection.

The standard option is to completely close the connection by passing in **0**. However, if you fork a process, make sure that the process which does not make PCM calls any more (usually the child process) closes all open file descriptors (FDs). You can do this by passing **1** as the value of **how**, which is **PCM_CONTEXT_CLOSE_FD_ONLY** in **pcm.h**. This allows the child process (in most cases) to close the FDs without closing the PCM connection in the parent process that spawned it. If you want the child process to continue making PCM calls, open another PCM connection.

ebufp
A reference to an error buffer obtained through pcm_perl_new_ebuf.

Return Values

This function returns nothing.

Error Handling

This function returns any errors to the error buffer.

pcm_perl_connect

This function connects to BRM by using PCM_CONNECT.

Syntax

```
pcm_context_t*
pcm_perl_connect(db_no, ebufp);
```

Parameters

db_no
The variable for the database number.

ebufp
A reference to an error buffer obtained through pcm_perl_new_ebuf.

Return Values

Returns an opaque reference to the PCM context and sets the database number to *db_no* if the function is successful.

Error Handling

This function returns any errors to the error buffer.

pcm_perl_context_open

This function opens a PCM context to BRM by using PCM_CONTEXT_OPEN.

Syntax

```
pcm_context_t*
pcm_perl_context_open(login_flistp, db_no, ebufp);
```

Parameters

login_flistp
A reference to the login flist. The login flist must have a dummy **PIN_FLD_POID**, a valid login type in **PIN_FLD_TYPE**, the **PIN_FLD_LOGIN**, and any other fields required for the given type, usually **PIN_FILD_PASSWD_CLEAR**. Connection Manager (CM) is declared in the **pin.conf** file or by one or more **PIN_FLD_CM_PTR** fields in the login flist.

db_no
The variable for the database number.

ebufp
A reference to an error buffer obtained through pcm_perl_new_ebuf.

Return Values

Returns an opaque reference to the PCM context and sets the database number to *db_no* if the function is successful.

Error Handling

This function returns any errors to the error buffer.

pcm_perl_destroy_ebuf

This function deletes a previously created error buffer from memory.

Syntax

```
void
pcm_perl_destroy_ebuf(ebufp);
```

Parameter

ebufp
A reference to the error buffer to be deleted.

Return Values

This function returns nothing.

Error Handling

This function does not handle errors.

pcm_perl_ebuf_to_str

This function returns a static string with a printable representation of the error buffer.

Syntax

```
char*
pcm_perl_ebuf_to_str(ebufp);
```

Parameter

ebufp
A reference to the error buffer.

Return Values

Returns a static string if the function is successful.

Error Handling

This function returns a null pointer if there are no errors or a printable string if there are errors.

pcm_perl_get_session

This function obtains the session ID set after login as a printable POID and returns it as a string.

Syntax

```
char*
pcm_perl_get_session(ctxp);
```

Parameter

ctxp
A reference to the open PCM context.

Return Values

Returns a printable string containing the session ID if the function is successful.

Error Handling

This function does not handle any errors.

pcm_perl_get_userid

This function obtains the user ID set after login as a printable POID and returns it as a string.

Syntax

```
char*
pcm_perl_get_userid(ctxp);
```

Parameter

ctxp
A reference to the open PCM context.

Return Values

Returns a printable string containing the user ID if the function is successful.

Error Handling

This function does not handle errors.

pcm_perl_is_err

This function checks for errors and returns the integer value of the error code in the error buffer.

Syntax

```
int
pcm_perl_is_err(erbufp);
```

Parameter

erbufp
A reference to the error buffer.

Return Values

Returns **0** if there are no errors. Returns the error code if there are errors.

Error Handling

This function returns the error code if an error occurred.

pcm_perl_new_ebuf

This function creates an empty error buffer structure and returns a pointer to it.

Syntax

```
pin_errbuf_t*
pcm_perl_new_ebuf();
```

Parameters

This function has no parameters.

Return Values

Returns a reference to the error buffer if the function is successful.

pcm_perl_op

This function performs the indicated PCM operation.

Syntax

```
pin_flist_t*
pcm_perl_op(ctxp, op, flag, in_flp, ebufp);
```

Parameters

ctxp
A reference to an open PCM context.

op
The PCM opcode that indicates the operation to be performed. *op* may be a number or symbolic opcode name, as long as it is known to BRM. For example, you can use **354** or **PCM_OP_TERM_IP_DIALUP_AUTHORIZE**.

For a list of opcode names, see PCM opcode libraries.

flag
A flag for the opcode. See the opcode description for information on the flags each opcode supports. Most opcodes take no flag, which is input as **(int32) 0**.

in_flp
A reference to the input flist.

For the input flist specifications, see PCM opcode libraries.

ebufp
A reference to the error buffer.

Return Values

Returns a reference to the resulting flist if the function is successful. Returns **NULL** if there is a serious error.

Note: You have to explicitly destroy both the input and return flists. They are not automatically deleted.

Error Handling

This function 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 application programming interface (API).

The following error codes returned from PCM_OP indicate an error in the Portal Communication Protocol (PCP) transmission:

- PIN_ERR_BAD_XDR
- PIN_ERR_STREAM_EOF
- PIN_ERR_STREAM_IO
- PIN_ERR_TRANS_LOST

- PIN_ERR_CM_ADDRESS_LOOKUP_FAILED

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

pcm_perl_print_ebuf

This function executes a **printf** of the printable representation of the error buffer.

Syntax

```
void
pcm_perl_print_ebuf(ebufp);
```

Parameter

ebufp
A reference to the error buffer to be printed.

Return Values

This function returns nothing.

Error Handling

This function prints the error buffer if there are errors. This function returns **pcm_perl_print_ebufp():NULL ptr** if there are no errors.

pin_flist_destroy

This function deletes an opaque flist.

Syntax

```
void
pin_flist_destroy(flistp);
```

Parameter

flistp
A reference to the flist to delete.

Return Values

This function returns nothing.

Error Handling

This function does not handle errors.

pin_flist_sort

This function sorts the specified flist using PIN_FLIST_SORT.

Syntax

```
void
pin_flist_sort(*flistp, *sort_flistp, reverse, sort_default, ebufp);
```

Parameters

flistp
A reference to the flist being sorted. The flist normally is an array and the sorting is performed on elements of the array. Each element of the array can be a list of fields; it is those fields that get sorted.

sort_listp
A list of fields in each element in *flistp* to use as sort fields. Elements in *flistp* are sorted in this order. If the value of this parameter is **NULL**, **PIN_ERR_BAD_ARG** is returned.

reverse
Reverses the order in which the flist is sorted.

sort_default
Compares nonexistent fields to existing fields.

For detailed information, see "PIN_FLIST_SORT".

ebufp
A reference to the error buffer.

Return Values

This function returns nothing.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series-style ebuf API routines by using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors.

pin_perl_flist_to_str

This function converts an opaque flist into a printable string representation.

For more information, see "PIN_FLIST_TO_STR".

Syntax

```
char*
pin_perl_flist_to_str(flistp, ebufp);
```

Parameters

flistp
A reference to the flist.

ebufp
A reference to the error buffer.

Return Values

Returns the flist in a printable string format if the function is successful. Returns **NULL** if the function fails.

Error Handling

This routine uses series-style ebuf error handling. Applications can call any number of series-style ebuf API routines by using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors.

For more information, see "Understanding API Error Handling and Logging" in *BRM Developer's Guide*.

pin_perl_str_to_flist

This function converts a printable flist into an opaque flist and returns a reference to the flist. If the flist uses the string '$DB_NO' for the database in the POID type fields, the value of *db_no* is substituted. In Perl, it is easier to set a variable $DB_NO and let Perl substitute the "DB_NO" if the flist is defined using **here** documents.

Syntax

```
pin_flist_t*
pin_perl_str_to_flist(str, db_no, ebufp);
```

Parameters

str
A reference to the destination string containing an flist in printable format.

db_no
A reference to the database number. Must be a string containing a BRM database number in dotted decimal format that is used to set the default database for parsing the flist.

ebufp
A reference to the error buffer.

Return Values

Returns the reference to the flist created from the input string if the function is successful. Returns **NULL** if the function fails.

Error Handling

This function uses series-style ebuf error handling. Applications can call any number of series-style ebuf API routines using the same error buffer and check for errors only once at the end of the series of calls. This makes manipulating flists and POIDs much more efficient because the entire logical operation can be completed and then tested once for any errors.

For more information, see "Understanding API Error Handling and Logging" in *BRM Developer's Guide*.

pin_perl_time

This function returns the time from the **pin_virtual_time** function, which is used to change time in BRM. You use this function for testing time-sensitive functions in BRM without affecting the system clock.

For more information, see "pin_virtual_time" in *BRM Developer's Guide*.

Syntax

```
time_t
pin_perl_time();
```

Parameters

This function has no parameters. However, for time offsets to take effect, there must be an entry for **pin_virtual_time** in the **pin.conf** file.

Return Values

Returns the time as a UNIX style time value: the number of seconds since 00:00:00 UTC, January 1, 1970.

Error Handling

This function does not handle errors.

pin_set_err

This function sets an error buffer.

Syntax

```
void
pin_set_err(ebufp, location, errclass, pin_err, field, recID, resvd);
```

Parameters

ebufp
A reference to the error buffer to be set.

location
The location of an error, which is one of the PIN_ERRLOC_*xxx*, where *xxx* indicates the subsystem that issued the error.

For details, see "pin_set_err".

errclass
One of the four classes of error PIN_ERRCLASS_*xxx*.

For details, see "pin_set_err".

pin_err
One of the system error messages PIN_ERR_*xxx*.

For details, see "pin_set_err".

field
Set this field to **0** or to the applicable PIN_FLD_*xxx*.

recilD
Set this field to **0** or to the record ID of the array element where the error occurred.

resvd
Reserved. Set this field to **0** or to a value chosen to provide further information about the specific error.

Return Values

This function returns nothing.

Error Handling

This function does not handle errors.

5

Storable Class-to-SQL Mapping

This chapter lists each Oracle Communications Billing and Revenue Management (BRM) storable class and the SQL tables to which it is mapped.

Storable Class-to-SQL Mapping

You use SQL directly with the database to generate reports. If you are an experienced system administrator, you can add indexes to improve performance. The default indexes are specified in the **create_indexes.source** file in the *BRM_Home*/**sys/dm_oracle/data/sql** directory.

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.
- Do not update or delete the default indexes.

SQL Mapping Matrix

A complete list of SQL tables and fields and their storable-class equivalents is in the file *BRM_Home*/**sys/dd/data/dd_objects.source**. Indexes are listed in the **create_indexes.source** file in the *BRM_Home*/**sys/dm_oracle/data/sql** directory.

For storable class-to-SQL mapping information, refer to the storable class descriptions. Each description includes the SQL mapping for every field in the class. See "Storable Class Definitions".

SQL Mapping Notes

When looking up SQL mapping indexes, keep in mind the following exceptions.

- The PIN_FLD_INTERNAL_NOTES field in the **/account** storable class is implemented by two fields in two separate tables: the field size is stored in the **/account** storable class as **internal_notes_size**, and the field value is stored in the table **account_internal_notes_buf**.
- The **PIN_FLD_BUFFER** field in the **/data** storable class is implemented by two fields in two separate tables: the field size is stored in the **/data** storable class as **buffer_size**, and the field value (the buffer) is actually stored in the table **data_buffer_buf**.

- SQL **recid** fields correspond to an element ID field.
- All **/event** storable subclasses inherit a set of fields from the **/event** super class, but they are implemented using different tables. The following **/event** storable subclasses are implemented using only the **event_t** table:
  - **/event/activity**
  - **/event/activity/admin**
  - **/event/billing/cycle/arrears**
  - **/event/billing/cycle/fold**
  - **/event/billing/cycle/forward**
  - **/event/billing/debit**
  - **/event/session/pcm_client**

  All other **/event** storable subclasses implemented using the **event_t** table plus one or more additional tables.
- All **/service/*** storable classes inherit a set of fields from the **/service** storable class. In addition, **/service/email** and **/service/pcm_client** are implemented using only the **service_t** table, and **/service/ip** and **/service/admin_client** each require an additional table.
- The **/data** storable class is a general data class that can be used to store any type of data, including blobs. Unless you have specifically created **/data** storable classes, you won't need to access them with SQL since they are generally not used by the system.

Doing SQL Joins

If POIDs (storable object IDs) are not being used as the join criteria, joins can be done with normal field comparisons.

If storable object IDs are being used to join tables (for example, to get information about an account and its current balances), simplified join criteria can be used. All tables have either POIDs, which are concatenations of five fields, or they have two-field storable object IDs, **obj_id0** and **obj_id1**. The **poid_id0** and **poid_id1** fields in the main tables (like **/account**, **/event**, and **/service**) are the same as the **obj_id0** and **obj_id1** fields in their related tables (that are used to implement arrays and substructures), respectively. For example:

```
poid_id0 in account_t = obj_id0 in account_balances_t
poid_id1 in account_t = obj_id1 in account_balances_t
```

The database number (**poid_db**) should be the same for all storable objects in the same database and you won't need to join on it. In most cases, just joining on the **poid_id0** and **poid_id1** fields are sufficient. The only case where this is not enough is in the case of array elements such as **/event** balance impacts where an SQL **rec_id** (or storable object element ID) is also required.

The **poid_rev** field is incremented each time a storable object is modified. This field should not be used or changed. It is not necessary as a join criteria.

rec_id fields are used to match on particular array elements.

Reserved Tables

The following storable objects/tables listed in Table 5–1 are found in *home*/**sys/data/sql/dd_objects.source** file are reserved for BRM use and should not be used by customers:

Table 5–1 Reserved Tables

Storable Object	Reserved SQL Table
/link	link_t
null object	access_table
/who	who_t

SQL Statement Information at Runtime

It is possible to obtain a list of SQL statements which correspond to an operation or sequence of events. See "Increasing the Level of Reporting for a DM" in *BRM System Administrator's Guide* for more details.

6

Event Notification Definitions

This chapter provides a brief description of each Oracle Communications Billing and Revenue Management (BRM) notification event and includes links to the notification event specifications. See "Using Event Notification" in *BRM Developer's Guide* for more information.

For more information about event notification definitions and field definitions, see *BRM Event Notification Reference*.

Event Notification Definitions

Table 6–1 lists the BRM event notification definitions and descriptions.

Table 6–1 Event Notification Definitions

Event Notification	Description
/event/billing/dispute/notify	Generated when an event is disputed. When the notification event is detected, BRM creates a reservation for the disputed amount to prevent misuse of resources during the dispute. For more information, see "Configuring Adjustments, Disputes, and Settlements" in *BRM Managing Accounts Receivable*.
/event/billing/settlement/notify	Generated when a dispute is settled. When the notification event is detected, BRM releases the reservation against the disputed amount as part of the settlement process. For more information, see "Configuring Adjustments, Disputes, and Settlements" in *BRM Managing Accounts Receivable*.
/event/notification	An abstract class to define event notifications.
/event/notification/account	An abstract class to define event notifications for operations on the account object.
/event/notification/account/create	Generated when an account is created.
/event/notification/account/delete	Generated when an account is deleted.
/event/notification/account/pre_delete	Generated at the start of the account deletion process.
/event/notification/activity	An abstract class to define event notifications on activities.
/event/notification/activity/out_of_order	Generated when an out-of-order event is detected. When the notification event is detected, Pipeline Manager automatically rerates events. For more information, see "About Automatic Rerating of Out-of-Order Events" in *BRM Setting Up Pricing and Rating*.
/event/notification/amt	An abstract class to define event notifications for operations on the Account Migration Manager process.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/amt/HoldCDRProcessing	Generated when Account Migration Manager begins migrating a group of accounts from one database schema to another. This event notifies the account-router Pipeline Manager that it needs to suspend all EDRs for the specified accounts. For more information, see "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
/event/notification/amt/MigrateAcct	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another. This event notifies the account-router Pipeline Manager that it needs to update the POIDs for the specified list of accounts. For more information, see "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
/event/notification/amt/MigrateDestination	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another. This event notifies the destination Pipeline Manager that it needs to update the account information stored in cache. For more information, see "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
/event/notification/amt/MigrateSource	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another. For more information, see "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
/event/notification/amt/ResumeCDRProcessing	Generated after both Account Migration Manager successfully migrates a group of accounts and all Pipeline Manager instances successfully update their account information. This event notifies the account-router Pipeline Manager that it can begin processing all suspended and new EDRs for the specified list of accounts. For more information, see "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
/event/notification/auto_rerate	Generated when an event is backdated and requires rerating. For more information, see "About Automatic Rerating of Backdated Events" in *BRM Setting Up Pricing and Rating*.
/event/notification/bal_grp	An abstract class to define event notifications for operations on the balance group object.
/event/notification/bal_grp/create	Generated when a new balance group is created.
/event/notification/bal_grp/modify	Generated when an existing balance group is modified.
/event/notification/billing	An abstract class to define event notifications for billing operations.
/event/notification/billing/end	Generated when final billing ends for an accounting cycle. **Note**: Final billing occurs after the end of the billing delay period, if configured.
/event/notification/billing/end_partial	Generated when partial billing ends for an accounting cycle. Partial billing occurs only if you configured delayed billing and you executed billing during the delay period.
/event/notification/billing/start	Generated when final billing starts for an accounting cycle. **Note**: Final billing occurs after the end of the billing delay period, if configured.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/billing/start_partial	Generated when partial billing starts for an accounting cycle. Partial billing occurs only if you configured delayed billing and you execute billing during the delay period.
/event/notification/customer	An abstract class to define event notifications for operations on the customer object.
/event/notification/customer/modify	Generated after an account is successfully modified.
/event/notification/customer/pre_modify	Generated just prior to an account modification.
/event/notification/customer/reg_complete	Generated when customer registration is complete.
/event/notification/customer/uniqueness_ confirmed	Generated after BRM confirms that a customer's account POID is unique for all database schemas in your multischema system.
/event/notification/cycle	An abstract class to define event notifications for cycle operations.
/event/notification/cycle/end	Generated at the end of a billing cycle either by the PCM_ OP_BILL_MAKE_BILL opcode or after applying the cycle fees.
/event/notification/cycle/start	Generated at the start of a billing cycle either by the PCM_ OP_BILL_MAKE_BILL opcode or before applying the cycle fees.
/event/notification/deal	An abstract class to define event notifications for operations on the deal object.
/event/notification/deal/change	When transitioning an account from one deal to another, this event is generated just prior to canceling the old deal.
/event/notification/deal/change_complete	When transitioning an account from one deal to another, this event is generated after successfully canceling the old deal.
/event/notification/deal/transition	When transitioning an account from one deal to another, this event is generated just prior to adding the new deal.
/event/notification/deal/transition_complete	When transitioning an account from one deal to another, this event is generated after successfully adding the new deal.
/event/notification/device	An abstract class to define event notifications for operations on the device object.
/event/notification/device/state	Generated after a device is successfully changed to a new state.
/event/notification/device/state/in_transition	Generated just prior to a device changing state.
/event/notification/offer_profile/create	Generated when a new offer profile is created.
/event/notification/offer_profile/delete	Generated when an offer profile is deleted.
/event/notification/offer_ profile/ThresholdBreach	Generated when the sum of current balance and the consumed reservation is equal to or greater than the nearest threshold configured in the offer profile for a service and resource ID.
/event/notification/offer_profile/update	Generated when an existing offer profile is modified.
/event/notification/order	An abstract class to define event notifications for operations on the order object.
/event/notification/order/state	Generated after an order is successfully changed to a new state.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/order/state/in_transition	Generated just prior to an order changing state.
/event/notification/plan	An abstract class to define event notifications for operations on the plan object.
/event/notification/plan/transition	Generated just prior to an account transitioning from one plan to another.
/event/notification/plan/transition_complete	Generated after an account is successfully transitioned to a new plan.
/event/notification/price	An abstract class to define event notifications for operations on the price object.
/event/notification/price/discounts	An abstract class to define event notifications for operations on the pricing discount object.
/event/notification/price/discounts/modify	Generated after a discount is created or updated in the BRM database. This event is used to synchronize discounts between BRM and external CRM applications.
/event/notification/price/products	An abstract class to define event notifications for operations on the pricing product object.
/event/notification/price/products/modify	Generated after a product is created or updated in the BRM database. This event is used to synchronize products between BRM and external CRM applications.
/event/notification/price/sponsorships	An abstract class to define event notifications for operations on the pricing sponsorship object.
/event/notification/price/sponsorships/modify	Generated after a **/sponsorship** object is created or updated in the BRM database. This event is used to synchronize sponsorship (chargeshare) data between BRM and external CRM applications.
/event/notification/price/tailormade_ products/create	Generated when a customized **/product** object is created. For more information, see "Modifying Rates and Price Models in a Product" in *BRM Managing Customers*.
/event/notification/price/tailormade_ products/modify	Generated when a customized **/product** object is modified. For more information, see "Modifying Rates and Price Models in a Product" in *BRM Managing Customers*.
/event/notification/process_audit	An abstract class to define event notifications for operations on the process audit object.
/event/notification/process_audit/create	Generated when Revenue Assurance Manager creates a **/process_audit** object. For more information, see "Understanding Revenue Assurance Manager" in *BRM Collecting Revenue Assurance Data*.
/event/notification/process_audit/update	Generated when Revenue Assurance Manager updates a **/process_audit** object with revenue assurance data. For more information, see "Understanding Revenue Assurance Manager" in *BRM Collecting Revenue Assurance Data*.
/event/notification/product/cancel/no_refund	Generated when a refund could not be applied due to a canceled override product. For more information, see "Configuring Event Notification for Override Pricing" in *BRM Setting Up Pricing and Rating*.
/event/notification/profile	An abstract class to define event notifications for operations on the profile object.
/event/notification/profile/create	Generated when a new profile is created.
/event/notification/profile/delete	Generated when a profile is deleted.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/profile/modify	Generated after a profile is successfully changed.
/event/notification/profile/pre_modify	Generated just prior to a profile being modified.
/event/notification/rate_change	Generated when a condition occurs that may require rerating. For more information, see "About Automatic Rerating" in *BRM Setting Up Pricing and Rating*.
/event/notification/ra_threshold	Generated by the **pin_ra_check_thresholds** utility when specified conditions for producing revenue leakage alerts occur. For more information, see "Setting Up Revenue Assurance Manager for Pipeline Batch Rating" in *BRM Collecting Revenue Assurance Data*.
/event/notification/rerating	An abstract class to define event notifications for the rerating operation.
/event/notification/rerating/end	Generated when a rerating job has finished.
/event/notification/rerating/PrepareToRerate	Generated just prior to the rerating process. This event notifies Pipeline Manager to suspend event data record (EDR) processing for all accounts affected by the rerating job.
/event/notification/rerating/ReratingCompleted	Generated after rerating completes successfully. This signals that Pipeline Manager should resume EDR processing for all accounts affected by the rerating job.
/event/notification/rerating/start	Generated just prior to the start of the rerating process. This signals that Pipeline Manager should halt EDR processing for all accounts affected by the rerating job.
/event/notification/rollover	An abstract class to define event notifications for the rollover operation.
/event/notification/rollover/end	Generated after a resource sub-balance is successfully rolled over to another cycle. For more information, see "About Rollovers" in *BRM Setting Up Pricing and Rating*.
/event/notification/rollover/start	Generated just prior to a resource sub-balance being rolled over from one cycle to another. For more information, see "About Rollovers" in *BRM Setting Up Pricing and Rating*.
/event/notification/rollover_correction	An abstract class to define event notifications for operations on the rollover correction object.
/event/notification/rollover_correction/rerate	Generated when a rollover correction during billing requires an event to be rerated. This rollover correction, in turn, is necessitated by delayed usage events after the end of the cycle. For more information, see "Enabling Rerating and Rollover Correction Due to Delayed Events" in *BRM Configuring and Running Billing*.
/event/notification/service	An abstract class to define event notifications for operations on the service object.
/event/notification/service_balgrp_transfer	An abstract class to define event notifications for the service balance group transfer operation.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/service_balgrp_transfer/data	Generated when either of the following occurs: ■ A service is transferred from one balance group to another. ■ A balance group is transferred from one bill unit to another. This is used to synchronize balance group transfer data between BRM and Pipeline Manager. For more information, see "About Transferring Services between Balance Groups" in *BRM Managing Accounts Receivable*.
/event/notification/service_balgrp_transfer/end	Generated after a service is successfully transferred from one balance group to another. For more information, see "About Transferring Services between Balance Groups" in *BRM Managing Accounts Receivable*.
/event/notification/service_balgrp_transfer/start	Generated just prior to a service being transferred from one balance group to another. For more information, see "About Transferring Services between Balance Groups" in *BRM Managing Accounts Receivable*.
/event/notification/service/create	Generated when a service is created.
/event/notification/service/delete	Generated when a service is deleted.
/event/notification/service/modify	Generated when a service is modified.
/event/notification/service/post_change	Generated after a service has been successfully updated.
/event/notification/service/pre_change	Generated just prior to a service being updated.
/event/notification/service/pre_create	Generated just prior to the creation of a service.
/event/notification/service/pre_purchase	Generated just prior to a product purchase.
/event/notification/suspense	An abstract class to define event notifications for the suspense operation.
/event/notification/suspense/batch_delete	Generated when a suspended batch is purged. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/batch_resubmit	Generated when a suspended batch is submitted for recycling. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/batch_writeoff	Generated when a suspended batch is written off. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/delete	Generated when a suspense record is deleted. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/edit	Generated when a suspense record is modified. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/recycle	Generated when a suspense record is recycled. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.
/event/notification/suspense/writeoff	Generated when a suspense record is written off. For more information, see "About Suspense Manager" in *BRM Configuring Pipeline Rating and Discounting*.

Table 6–1 (Cont.) Event Notification Definitions

Event Notification	Description
/event/notification/svc_order	An abstract class to define event notifications for operations on the service order object.
/event/notification/svc_order/state	Generated after a service order is successfully changed to a new state.
/event/notification/svc_order/state/in_transition	Generated just prior to a service order changing state.
/event/notification/threshold	Generated when an balance crosses *above* a threshold value or credit limit. For more information, see "Alerting Customers When Monitored Balances Cross Limits or Thresholds" in *BRM Managing Accounts Receivable* and "About Credit Limit and Threshold Checking during Batch Rating" in *BRM Managing Customers*.
/event/notification/threshold_below	Generated when a balance crosses *below* a threshold value or credit limit. For more information, see "Alerting Customers When Monitored Balances Cross Limits or Thresholds" in *BRM Managing Accounts Receivable* and "About Credit Limit and Threshold Checking during Batch Rating" in *BRM Managing Customers*.

7

Pipeline Manager iScript Functions

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iScript functions.

For information on creating custom iScript and iRules modules, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

Arithmetic Functions

Table 7–1 contains the arithmetic functions.

Table 7–1 Arithmetic Functions

Function	Description
decimalAbs	Derives an absolute value from a decimal value.
decimalToLong	Converts the integer portion of a decimal value to a Long value.
longAbs	Derives an absolute value from a Long value.
longToDecimal	Converts a Long value to a decimal value.
round	Rounds a decimal value to a specified number of decimal places.
sqrt	Calculates the square root of the input value.
trunc	Truncates a decimal value to a specified number of decimal places.

decimalAbs

This function derives an absolute value from a decimal value.

Syntax

```
Decimal decimalAbs(Decimal source);
```

Parameter

source
The decimal value from which to derive the absolute value.

Return Values

Returns the derived absolute value.

Example

```
if ( x == decimalAbs( x ) )
{
  logFormat( "x is a positive value" );
}
```

decimalToLong

This function converts the integer portion of a decimal value to a Long value.

Syntax

```
Long decimalToLong(Decimal source);
```

Parameter

source
The decimal value to convert to a Long value.

Return Values

Returns the Long value of the integer portion of the decimal value.

Example

```
Long p = decimalToLong( 3.1415 );
```

longAbs

This function derives an absolute value from a Long value.

Syntax

```
Long longAbs(Long source);
```

Parameter

source
The Long value from which to derive the absolute value.

Return Values

Returns the derived absolute value.

Example

```
if ( x == longAbs( x ) )
{
  logFormat( "x is a positive value" );
}
```

longToDecimal

This function converts a Long value to a decimal value.

Syntax

```
Decimal longToDecimal(Long value);
```

Parameter

value
The Long value to convert to a decimal value.

Return Values

Returns the converted decimal value.

Example

```
Decimal bytesPerSecond = longToDecimal( bytes ) / \
longToDecimal( seconds );
```

round

This function rounds a decimal value to a specified number of decimal places.

Syntax

```
Decimal round(Decimal value [, Long places] [, String mode]);
```

Parameters

value
The value to round.

places
The number of decimal places to achieve when rounding, also known as the number of significant digits (the default is **0**).

mode
The rounding mode, or method of rounding. Possible values:

- **ROUND_PLAIN:** (Default) If the digit following the last significant digit is 5 or greater, round up. If the digit following the last significant digit is less than 5, round down.
- **ROUND_UP:** Always round up if the digit following the last significant digit is greater than 0.
- **ROUND_DOWN:** Always round down. This is the same as truncating all digits following the last significant digit.
- **ROUND_BANKERS:** This mode rounds in one of the following ways, depending on the value of the digit following the last significant digit:
  - If it is less than 5, truncate all digits following the last significant digit.
  - If it is greater than 5, round up.
  - If it is 5, round to the nearest even digit. For example, if the precision is 2, 10.155 and 10.165 both round to 10.16 because 6 is an even number.

Return Values

Returns the value rounded to the specified decimal place.

Example

```
Decimal r = round( 3.1415, 3 ); // r now is 3.142
```

sqrt

This function calculates the square root of the input value.

Syntax

```
Decimal sqrt(Decimal value);
```

Parameter

value
The value for which to calculate the square root.

Return Values

Returns the square root of the input value.

Example

```
Decimal c = sqrt( a*a + b*b );
```

trunc

This function truncates a decimal value to a specified number of decimal places.

Syntax

```
Decimal trunc(Decimal value [, Long places]);
```

Parameters

value
The value to truncate.

places
The number of decimal places by which the value should be truncated (the default is **0**).

Return Values

Returns the value truncated to the specified decimal place.

Example

```
Decimal t = trunc( 3.1415, 3 ); // t now is 3.141
```

ASN.1 Functions

Table 7–2 contains the ASN.1 functions.

Table 7–2 ASN.1 Functions

Function	Description
asnTreeAddInteger	Adds an integer object to the current active node of the ASN tree.
asnTreeAddString	Adds a string object to the current active node of the ASN tree.
asnTreeCreate	Creates a tree in memory to hold an ASN.1 file structure.
asnTreeDelete	Deletes the last created or used ASN.1 tree.
asnTreeDeleteNodeByIndex	Deletes a node from the ASN.1 tree.
asnTreeFlush	Flushes the content of the ASN.1 tree to the output.
asnTreeGetStoredNode	Gets the active (working) node from a list of created and store
asnTreePop	Backs up one level in the ASN.1 tree hierarchy.
asnTreePushTag	Adds a new block to the current active node of the ASN.1 tree and sets this new block as an active node of the tree.
asnTreeStoreNode	Stores an index to a constructed block node in the ASN.1 tree, when the block position in the tree is fixed.

asnTreeAddInteger

This function adds an integer object to the current active node of the ASN.1 tree.

Syntax

```
Bool asnTreeAddInteger(String blockName, Long value);
```

Parameters

blockName
The name of the block to add (exact type from the block description file).

value
The integer to insert as the value.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeAddInteger("TAP3.DataVolume.DATA_VOLUME", 2512);
...
```

asnTreeAddString

This function adds a string object to the current active node of the ASN.1 tree.

Syntax

```
Bool asnTreeAddString(String blockName, String value)
```

Parameters

blockName
The name of the block to add. This must exactly match the type from the block description file.

value
The string to insert as the value.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeAddString("TAP3.CalledPlace.CALLED_PLACE", "Freephone");
...
```

asnTreeCreate

This function creates a tree in memory to hold an ASN.1 file structure, where the Length field of the objects can be calculated in the end, just before writing on the output.

Parameters

None.

Return

True on success, otherwise, **False**

Only one tree can be in use at a time.

Example

```
...
if ( asnTreeCreate() == false )
{
logFormat( "asnTreeCreate() failed.");
}
...
```

asnTreeDelete

This function deletes the last created or used ASN.1 tree.

Syntax

```
Bool asnTreeDelete();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise returns **False**.

Example

```
...
if ( asnTreeDelete() == false )
{
logFormat( "asnTreeDelete() failed.");
}
...
```

asnTreeDeleteNodeByIndex

This function deletes a node from the ASN.1 tree, by recursively deleting all contained blocks and values.

Syntax

```
Bool asnTreeDeleteNodeByIndex(Long nodeIdx);
```

Parameter

nodeIdx
Node index in the ASN.1 tree as returned by asnTreeStoreNode().

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
//there is no need for this optional block (no data to store
//in it), so delete it
asnTreeDeleteNodeByIndex(networkInfoIdx);
...
```

asnTreeFlush

This function flushes the content of the ASN.1 tree to the output.

Syntax

```
Bool asnTreeFlush();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
if ( asnTreeFlush() == false )
{
logFormat( "asnTreeFlush() failed.");
}
...
```

asnTreeGetStoredNode

This function gets the active (working) node from a list of created and stored, but not filled, constructed blocks.

Syntax

```
Bool asnTreeGetStoredNode(Long nodeIdx);
```

Parameter

nodeIdx
Node index in the ASN.1 tree as returned by asnTreeStoreNode().

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeGetStoredNode(networkInfoIdx);
//use asnTreeAddString() and asnTreeAddInteger() to update
//the TAP3.NetworkInfo block.
...
```

asnTreePop

This function backs up one level in the ASN.1 tree hierarchy. Every asnTreePushTag(xxxx) should have an associated asnTreePop(); it is like opening and closing brackets.

Syntax

```
Bool asnTreePop();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreePushTag("TAP3.AuditControlInfo");
...
asnTreePop(); //asnTreePushTag("TAP3.AuditControlInfo");
...
```

asnTreePushTag

This function adds a new block to the current active node of the ASN.1 tree and sets this new block as an active node of the tree. Use this function to create constructed ASN.1 objects, for example, **SEQUENCE** or **CHOICE**.

If the **isIndefiniteLength** parameter is set to true, the Length field of the ASN.1 object is set to 0x80 and 2 null bytes are appended to the Value field of the ASN.1 object.

Syntax

```
Bool asnTreePushTag(String blockName [, Bool isIndefiniteLength=false] );
```

Parameters

blockName
The name of the structured block to add (exact type from the block description file).

isIndefiniteLength
Flag to indicate that the generated block has to use indefinite lengths. The default is false, that is, it stores the exact size of the value field in the objects header.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreePushTag("TAP3.AuditControlInfo");
...
```

asnTreeStoreNode

This function stores an index to a constructed block node in the ASN.1 tree, when for example, the data values that should be put in this block are unknown, but the block position in the tree is fixed.

Syntax

```
Long asnTreeStoreNode();
```

Parameters

None.

Return Values

Returns a node index that can be used with asnTreeGetStoredNode(nodeIdx) or asnTreeDeleteNodeByIndex(nodeIdx).

Example

```
...
asnTreePushTag("TAP3.NetworkInfo");
Long networkInfoIdx = asnTreeStoreNode();
//Nothing to do now, node will be updated after all //details are processed
asnTreePop(); //for asnTreePushTag("TAP3.NetworkInfo");
...
```

The following example iScript demonstrates how to create an output file in ASN.1 containing only a list of QoS requested objects (one per event data record (EDR)), with all field values set to **3**.

This is the content of the **OutGrammar.dsc** file. There should be an associated file describing the block structure that is here used, for example, **TAP3.QoSRequestedList**.

```
// The initial iScript code
iScript
{
  use EXT_AsnTree; // iScript extension to build a Tree of ASN.1 object
                   // used to fill the Length value of the ASN.1 bloc,
                   // before printing on output stream
}
// The definition of the grammar
Grammar
{
  edr_stream:
    header
    details
    trailer
  ;
  header:
    HEADER
    {
      asnTreeCreate();
      asnTreePushTag("TAP3.QoSRequestedList");
    }
  ;
  trailer:
    TRAILER
```

```
    {
      asnTreePop(); //for asnTreePushTag("TAP3.QoSRequestedList");
      asnTreeFlush();
      asnTreeDelete();
    }
  ;
  details:
    details
    DETAIL
    {
      asnTreePushTag("TAP3.QoSRequested");
      asnTreeAddInteger("TAP3.QoSDelay.QOS_DELAY", 3);
      asnTreeAddInteger("TAP3.QoSMeanThroughput.QOS_MEAN_THROUGHPUT", 3);
      asnTreeAddInteger("TAP3.QoSPeakThroughput.QOS_PEAK_THROUGHPUT", 3);
      asnTreeAddInteger("TAP3.QoSPrecedence.QOS_PRECEDENCE", 3);
      asnTreeAddInteger("TAP3.QoSReliability.QOS_RELIABILITY", 3);
      asnTreePop(); //for asnTreePushTag("TAP3.QoSRequested");
    }
  | /*EMPTY*/
  ;
}
```

Database Connection Functions

Table 7–3 contains database connection functions.

Table 7–3 Database Connection Functions

Function	Description
dbBeginTransaction	Starts a new transaction using the specified connection.
dbCloseConnection	Closes a connection to the Pipeline Manager database.
dbCloseResult	Closes a result handle after processing the result data.
dbCommitTransaction	Commits a transaction to a specific connection.
dbConnection	Establishes a connection to the Pipeline Manager database. The handle returned by this function should be used in future calls to the **dbExecute** function.
dbDataConnection	Connects the extension to a DBC_Database module.
dbError	Retrieves a description for the last error. This description is not reset after a valid call to one of the other database connection functions. Therefore, **dbError** should only be called directly after one of the other database connection functions fails.
dbExecute	Executes an SQL statement against the Pipeline Manager database.
dbNextResult	Switches the cursor to the next result for the result handle you specify.
dbNextRow	Switches the cursor to the next row in the current result.
dbRollbackTransaction	Rolls the current transaction back for the specified connection.

dbBeginTransaction

This function starts a new transaction using the specified connection.

Syntax

```
Bool dbBeginTransaction(Long conHandle);
```

Parameter

conHandle
The connection you want to use for the new transaction.

Return Values

Returns **true** if the transaction was successfully started. Returns **false** if the function fails.

Example

```
if ( dbBeginTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to begin a new transaction: " \
  + dbError() );
}
```

dbCloseConnection

This function closes a connection to the Pipeline Manager database.

Syntax

```
Bool dbCloseConnection(Long conHandle);
```

Parameter

conHandle
The connection you want to close.

Return Values

Returns **true** if the connection was successfully closed. Returns **false** if the function fails.

Example

```
if ( dbCloseConnection( conHandle ) == false )
{
  logFormat( "ERROR: failed to close a connection: " + \
  dbError() );
}
```

dbCloseResult

This function closes a result handle after processing the result data.

Syntax

```
Bool dbCloseResult(Long resHandle);
```

Parameter

resHandle
The result handle you want to close.

Return Values

Returns **true** if the result handle was successfully closed. Returns **false** if the function fails.

Example

```
resHandle = dbExecute( "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
...

// Process the result data

...
dbCloseResult( resHandle );
```

dbCommitTransaction

This function commits a transaction to a specific connection.

Syntax

```
Bool dbCommitTransaction(Long conHandle);
```

Parameter

conHandle
The connection you want to use for the transaction.

Return Values

Returns **true** if the transaction was successfully committed to the connection. Returns **false** if the function fails.

Example

```
if ( dbCommitTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to commit the transaction: " + dbError() );
}
```

dbConnection

This function establishes a connection to the Pipeline Manager database. The handle returned by this function should be used in future calls to the **dbExecute** function.

> **Note:** Before calling **dbConnection**, connect to DBC_Database module using **dbDataConnection**.

Syntax

```
Long dbConnection();
```

Parameters

None.

Return Values

Returns the handle for the new connection (the handle is a value greater than or equal to 0) if the function is successful. Returns **INVALID_CONNECTION** if the function fails.

Example

```
conHandle = dbConnection();
if ( conHandle == INVALID_CONNECTION )
{
  logFormat( "ERROR: dbConnection() failed: " + dbError() );
}
```

dbDataConnection

This function connects the extension to a DBC_Database module. This connection is valid for the whole extension; you cannot connect the extension to two different DBC_Database modules.

> **Note:** Before calling **dbConnection**, connect to DBC_Database module using **dbDataConnection**.

Syntax

```
Bool dbDataConnection(String dbcModule);
```

Parameter

dbcModule
The registry name for the DBC_Database module.

Return Values

Returns **true** if the extension was successfully connected to the module. Returns **false** if the function fails.

Example

```
use IXT_Db;

if ( dbDataConnection( "integrate.DataPool.Login.Module" ) == \
true )
{
  logFormat( "Connection to DBC module established" );
}
else
{
  logFormat( "ERROR: failed to establish the connection \
  to DBC module" );
}
```

dbError

This function retrieves a description for the last error. This description is not reset after a valid call to one of the other database connection functions. Therefore, **dbError** should only be called directly after one of the other database connection functions fails.

Syntax

```
String dbError();
```

Parameters

None.

Return Values

Returns a description of the error.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
```

dbExecute

This function executes an SQL statement against the Pipeline Manager database. The handle this function returns should be used to access the result of the SQL statement in the **dbNextResult** and **dbNextRow** calls that follow. After processing the result data, free the handle by calling **dbCloseResult**.

Syntax

```
Long dbExecute(Long conHandle, String sqlStatement);
```

Parameters

conHandle
The connection you want to use.

sqlStatement
The SQL statement to execute.

Return Values

Returns the result handle (the handle is a value greater than or equal to 0) if the function is successful. Returns **INVALID_RESULT** if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
```

dbNextResult

This function switches the cursor to the next result for the result handle you specify.

> **Note:** This function is specific to results, not rows. The return generated by **dbExecute** can consist of a list of results in table form, with each result containing one or more data rows. Using **dbNextResult** moves the cursor from result to result, not from data row to data row within a result.

Syntax

```
Long dbNextResult(Long resHandle);
```

Parameter

resHandle
The result handle you want to process.

Return Values

Returns the next result in the result handle if the function is successful. Returns **NO_MORE_RESULTS** if the function reaches the last result. Returns a value less than 0 if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );

// loop for all results
do
{
  // process the rows of the current result
  while ( (ret = dbNextResult( resHandle )) == NEXT_RESULT );

if ( ret != NO_MORE_RESULTS )
{
  logFormat( "ERROR: dbNextResult() failed: " + dbError() );
}
```

dbNextRow

This function switches the cursor to the next row in the current result.

> **Note:** This function is specific to rows, not results. The return generated by **dbExecute** can consist of a list of results in table form, with each result containing one or more data rows. Using **dbNextRow** moves the cursor from row to row within a result, not from result to result.

Syntax

```
Long dbNextRow(Long resHandle, ...);
```

Parameters

resHandle
The handle for the result you want to process.

A list of bound variables

Return Values

Returns the next row in the result if the function is successful. Returns **NO_MORE_ROWS** if the function reaches the last row. Returns a value less than 0 if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );

// loop for all rows
while ( (rowRet = dbNextRow( resHandle, cli, validFrom validTo )) > 0 )
{
  ...
}

if ( rowRet != NO_MORE_ROWS )
{
  logFormat( "ERROR: dbNextRow() failed: " + dbError() );
}
```

dbRollbackTransaction

This function rolls the current transaction back for the specified connection.

Syntax

```
Bool dbRollbackTransaction(Long conHandle);
```

Parameter

conHandle
The connection whose transaction you want rolled back.

Return Values

Returns **true** if the rollback is successful. Returns **false** if the function fails.

Example

```
if ( dbRollbackTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to rollback current transaction: " \
  + dbError() );
}
```

Data Normalizing Functions

Table 7–4 contains data normalizing functions.

Table 7–4 Data Normalizing Functions

Function	Description
convertCli	Normalizes wireline and wireless command-line interfaces (CLIs). Static class function: "EXT_ConvertCli::convert"
convertIPv4	Normalizes IPv4 addresses. Static class function: "EXT_ConvertIPv4::convert"
String convertIPv6	Normalizes IPv6 addresses. Static class function: "EXT_ConvertIPv6::convert"
convertIPv4onv6	Normalizes IPv4 over IPv6 addresses. Static class function: "EXT_ConvertIPv4onv6::convert"

convertCli

This function normalizes wireless and wireline CLIs into international format.

Syntax

```
String convertCli( String cli,
                   String modInd,
                   Long typeOfNumber,
                   String natAccessCode,
                   StringArray intAccessCode,
                   StringArray countryCode,
                   String intAccessCodeSign,
                   String natDestinCode )
```

Parameters

cli
CLI to normalize.

modInd
Modification Indicator, for example, "00".

typeOfNumber
Type Of Number, for example, 0.

natAccessCode
National Access Code, for example, "0".

intAccessCode
International Access Code, for example, "00".

countryCode
Country Code, for example, "49".

intAccessCodeSign
International Access Code Sign, for example, "+".

natDestinCode
National Destination Code, for example, "172".

Return Values

Returns a CLI in international normalized format: <iac>< cc><ndc>extension.

Example

```
...
use EXT_Converter;

String normCli;
String cli = "01721234567";

normCli = convertCli( cli, "00", 0, "0", "00", "49", "+", "172" );

// normCli now contains: 00491721234567

...
```

convertIPv4

This function normalizes IPv4 addresses.

Syntax

```
String convertIPv4( String ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 3 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "192.168.1.253";

normIp = convertIPv4( ip );

// normIp now contains: 192168001253

...
```

String convertIPv6

This function normalizes IPv6 addresses.

Syntax

```
String convertIPv6(String ip;
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "0:0:0:AF:E:0:1:FE";

normIp = convertIPv6( ip );

// normIp now contains: 00000000000000AF000E0000000100FE

...
```

convertIPv4onv6

This function normalizes IPv4 over IPv6 addresses. The decimal IPv4 address is converted into hexadecimal representation.

Syntax

```
String convertIPv4onv6(String ip);
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IPv6 address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "0:0:0:0:0:0:192.168.10.1";

normIp = convertIPv4onv6( ip );

// normIp now contains: 00000000000000000000000C0A80A01

...
```

Date Functions

Table 7–5 contains date functions.

Table 7–5 Date Functions

Function	Description
dateAdd	Adds date and time values.
dateDiff	Calculates the difference between two dates.
dateIsValid	Checks a date for validity; for example, after initialization from a string.
dateToStr	Converts a date value to a string.
strToDate	Converts a string into a date value.
sysdate	Retrieves the current system date.

dateAdd

This function manipulates date and time values.

Syntax

```
Date dateAdd(Date source [, Long years [, Long months [, days [, Long hours [,
Long mins [, Long secs]]]]]]);
```

Parameters

source
The source date for the addition.

years
The number of years to add. This parameter can be positive or negative.

months
The number of months to add. This parameter can be positive or negative.

days
The number of days to add. This parameter can be positive or negative.

hours
The number of hours to add. This parameter can be positive or negative.

mins
The number of minutes to add. This parameter can be positive or negative.

secs
The number of seconds to add. This parameter can be positive or negative.

Return Values

Returns the manipulated source date.

Note: The variable source itself is not manipulated; only the result is returned.

Example

```
Date now = sysdate();
Date later = dateAdd( now, 1, 2, 0, 5 );

logStdout( "Date now is " + dateToStr(now) + "\n" );
logStdout( "In 1 year, 2 months and 5 hours it is " + dateToStr(later) + "\n" );
```

dateDiff

This function calculates the difference between two dates. The difference is returned in seconds.

Syntax

```
Long dateDiff(Date date1, Date date2);
```

Parameters

date1
The first date used for calculating the difference. This is the minuend.

date2
The second date used for calculating the difference. This is the subtrahend.

Return Values

Returns the difference between the first and second date, in seconds.

Example

```
if ( dateDiff( sysdate(), date ) < 0 )
{
  logFormat( "the date is a future date" );
}
```

dateIsValid

This function checks a date for validity; for example, after initialization from a string.

Syntax

```
Bool dateIsValid(Date date);
```

Parameter

date
The date to validate.

Return Values

Returns **true** if the date is valid. Returns **false** if the date is not valid.

Example

```
Date timeStamp = strToDate( timeString );
if ( dateIsValid( timeStamp ) == false )
{
  logFormat( timeString + " is no valid date string" );
}
```

dateToStr

This function converts a date value to a string.

Syntax

```
String dateToStr(Date date);
```

Parameters

%a
The abbreviated week day name; for example, Sun for Sunday. This is from tm::tm_wday.

%A
The full weekday name; for example, Sunday. This is from tm::tm_wday.

%b
The abbreviated month name; for example, Feb for February.

%B
The full month name; for example, February.

%c
The date and time; for example, Feb 29 14:34:56 2004. This may use all members.

%d
The day of the month; for example, 29.

%H
The hour of the 24-hour day; for example, 14.

%I
The hour of the 12-hour day; for example, 02.

%j
The day of the year starting from 001; for example, 060. This is from tm::tm_yday.

%m
The month of the year, from 01; for example, 02.

%M
The minutes after the hour; for example, 34.

%p
The AM/PM indicator, if any; for example, AM.

%S
The seconds after the minute; for example, 56.

%U
The week of the year, starting from 00; for example, 45. This is from tm::tm_yday and tm::tm_wday. The week is defined as starting on Sunday.

%w
The day of the week, with 0 for Sunday; for example, 2 for Tuesday.

%W
The week of the year, from 00; for example, 33. This is from tm::tm_yday and tm::tm_wday. In this case, the week is defined as starting on Monday.

%x
The date; for example, Feb 29 2004. This uses tm::tm_yday in some locales.

%X
The time; for example, 14:34:56.

%y
The year of the century, from 00; for example, 04 for 2004. In most cases, you should avoid this parameter; to ensure correct handling of the past century, use **%Y** instead.

%Y
The year including the century; for example, 1994.

%Z
The time zone name; for example, PST or PDT. This is from tm::tm_isdst.

Return Values

Returns the date as a string using the format defined by the function parameters if the function is successful. Returns an empty string if the date is invalid.

Example

```
dateToString("%a %d. %B %Y")
```

will result in:

```
"Mon 24. June 2002"
```

strToDate

This function converts a string into a date value. The only supported string format is *YYYYMMDDHHMMSS*.

Syntax

```
Date strToDate(String dateStr);
```

Parameters

%%
The literal % character.

%d
The day of the month; for example, 29. The range is 00-31.

%H
The hour of the 24-hour day; for example, 14. The range is 00-23.

%m
The month of the year, from 01; for example, 02. The range is 01-12.

%M
The minutes after the hour; for example, 34. The range is 00-59.

%S
The seconds after the minute; for example, 56. The range is 00-59.

%y
The year of the century, from 00; for example, 04 for 2004. The range is 01-99. In most cases, you should avoid this parameter.

%Y
The year including the century; for example, 1994.

Return Values

Returns a valid date if the input string is in the right format. Returns an invalid date if the format is not correct.

Example

```
edrDate(DETAIL.CHARGING_START_TIMESTAMP) = \
strToDate("24.12.2002", "%d. %m. %Y");
```

sysdate

This function retrieves the current system date.

Syntax

```
Date sysdate();
```

Parameters

None.

Return Values

Returns the current system date.

Example

```
Date now;
now = sysdate();
```

EDR Container Functions

Table 7–6 contains EDR container functions.

Table 7–6 EDR Container Functions

Function	Description
edrAddAdditionalStream	Adds additional output streams to each EDR.
edrAddDatablock	Adds a new data block to the current EDR container.
edrAddDatablockEx	Adds a new data block to the current EDR container.
edrAddError	Adds a new error to the current EDR container.
edrArrayIndex	Accesses the array index values in EDR container.
edrClearErrors	Clears the list of errors that the pipeline modules add to the EDR container.
edrConnectToken	Associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field. This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.
edrConnectTokenEx	Associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field. This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.
edrContainsAdditionalStream	Determines whether an EDR has an additional output stream with the name you pass in.
edrCurrentTokenIndex	Provides the index of the token parsed from the stream. Valid only in input grammar.
edrDate	Retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values.
edrDateEx	Retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values.
edrDecimal	Retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values.
edrDecimalEx	Retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values.
edrDelete	Deletes the current EDR container, changing the current pointer to the EDR container directly in front of the deleted EDR.
edrDeleteDatablock	Deletes a data block from the current EDR container. The function is not supported for nested transactions.
edrDeleteField	Clears the contents of a field in an EDR container. The function is not supported for nested transactions.
edrDuplicate	Duplicates the current EDR container.

Table 7–6 (Cont.) EDR Container Functions

Function	Description
edrEmptyInput	Sets the state of a field to **EDR_INPUT_EMPTY** when the field is present in the CDR but contains no value.
edrFieldConnectInfo	Retrieves the Info string associated with the token for the corresponding EDR field. By default, the Info string contains the description of the token type. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
edrFieldTokenBytePos	Calculates the position of the token associated with the corresponding EDR field. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
edrGetAdditionalStream	Gets the name of an additional EDR output stream given an array index number.
edrGetError	Retrieves the names of the attached error messages.
edrGetErrorParameters	Retrieves the parameters associated to a specified error.
edrGetErrorSeverity	Retrieves the severity for each of the associated errors.
edrGetStream	Gets the output stream for an EDR.
edrHasError	Retrieves the names of the attached error messages.
edrInputState	Retrieves the input state of an EDR field.
edrInternalState	Returns the internal state of an EDR field.
edrInternalStateEx	Returns the internal state of an EDR field.
edrIsValidDetail	Determines whether the current EDR container is a valid detail container.
edrLong	Retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values.
edrLongEx	Retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values.
edrMaxSeverity	Finds the maximum severity of the errors added to the current EDR container.
edrMissingInput	Sets the state of a field to **EDR_INPUT_MISSING** when the field is not present in the CDR.
edrNumDatablocks	Determines the number of data blocks of the specified type.
edrNumDatablocksEx	Determines the number of data blocks of the specified type.
edrNumErrors	Accesses the number of error messages attached to the current EDR container.
edrNumTokens	Accesses the number of tokens attached to the current EDR container.
edrRemoveAdditionalStream	Removes additional output streams from an EDR.
edrSetContentType	Sets the content type of the current EDR container.
edrSetCurrent	Sets the current EDR container.
edrSetIsValidDetail	Sets the EDR container's valid detail flag. The valid detail flag specifies whether the EDR container is to be discarded.

Table 7–6 (Cont.) EDR Container Functions

Function	Description
edrSetStream	Sets the output stream for an EDR.
edrString	Retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values.
edrStringEx	Retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values.
edrTokenString	Used to retrieve the content of each token, as identified by their indexes. When the index is not available, as for a function call with no argument, this function returns the complete byte string attached to the EDR. The byte string corresponds to the original input string that generated the EDR. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
iRulesModeOn	Enables the iRules mode.
iRulesModeOff	Disables the iRules mode.
pipelineName	Retrieves the name of the pipeline in which the script is running.
stopPipeline	Stops the pipeline from which it is called.

edrAddAdditionalStream

This function adds additional output streams to each EDR.

Each Out_GenericStream pipeline module has a default output stream for EDRs. You use this function to add additional output streams to direct the output to additional locations.

Output stream characteristics (output path, record prefix, and record suffix) are set in the registry file.

If the stream name sent in with this function already exists, **edrAddAdditionalStream** returns **true** but does not create the stream again.

Syntax

```
Bool edrAddAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the new output stream that you are adding.

Return Values

Returns **true** if the function is successful. Returns **false** for all other conditions.

Example

This iScript example adds two additional output module streams:

```
addoutmod.isc
------
function onDetailEdr
{
      if (edrAddAdditionalStream( "TElOut1" ) == true)
   {
      logStdout("Stream TelOut1 added ");
   }
      if (edrAddAdditionalStream( "TELOut2" ) == true)
   {
      logStdout("Stream TElOut2 added ");
   }
} // end onDetailEdr + end iScript ----
```

This registry fragment shows the two example iScript files, **addoutmod.isc** and **removeoutmod.isc**, defined in the FunctionPool section. These iScripts add and remove output module streams. The new iScripts are shown in **bold**.

```
   FunctionPool
   {
      Iscript
      {
         ModuleName = FCT_IScript
         Module
         {
            Active = True
            Source = FILE
            Scripts
```

```
            {
               addoutmod
               {
                  FileName = ./samples/simple/addoutmod.isc
               }
            }
         }
         Iscript2
         {
            ModuleName = FCT_IScript
            Module
            {
               Active = True
               Source = FILE
               Scripts
               {
                  removeoutmod
                  {
                     FileName = ./samples/simple/removeoutmod.isc
                  }
               }
            }
         }
```

This output registry section defines the **TELOut1** output section:

```
TELOut1
{
   ModuleName = OUT_GenericStream
   Module
   {
      Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_OutGrammar.dsc
      DeleteEmptyStream = TRUE
      OutputStream
      {
         ModuleName = EXT_OutFileManager
         Module
         {
            OutputPath = ./samples/simple/data/out2
            OutputPrefix = Sol42_
            OutputSuffix = .out

            TempPrefix = tmp
            TempDataPath = ./samples/simple/data/out2
            TempDataPrefix = out.tmp.
            TempDataSuffix = .data

            Replace = TRUE
         }
      }
   }
}
```

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputPrefix, and OutputSuffix values for each output stream defined in the registry.

edrAddDatablock

This function adds a new data block to the current EDR container.

Syntax

```
Bool edrAddDatablock(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The name of the EDR block you want to add.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrAddDatablock( DETAIL.ASS_CBD ) == false )
{
  logFormat( "ERROR: failed to add ASSOCIATED_CHARGE \
  datablock" );
}
```

edrAddDatablockEx

This function adds a new data block to the current EDR container.

Syntax

```
Bool edrAddDatablockEx(String block, Long indicesArray, Long numIndices);
```

Parameters

block
The name of the EDR block you want to add.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD";
numberOfIndices = 0;

if ( edrAddDatablockEx(edrFieldName, indicesArray, numberOfIndices) == false )
{
  logFormat( "ERROR: failed to add ASSOCIATED_CHARGE \
  datablock" );
}
```

edrAddError

This function adds a new error to the current EDR container.

Syntax

```
Bool edrAddError(String error, Long severity [, String paramX...]);
```

Parameters

error
The name of the error you want to add to the EDR container.

severity
The severity of the error:

- **0** = Debug
- **1** = Normal
- **2** = Warning
- **3** = Minor error
- **4** = Major error
- **5** = Critical error

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrString( DETAIL.SERVICE_CODE ) != "Tel" and \
edrString( DETAIL.SERVICE_CODE ) != "Fax" )
{
  edrAddError( "ERR_UNKNOWN_SERVICE_CODE", 3, edrString\
  ( DETAIL.SERVICE_CODE ) );
}
```

edrArrayIndex

This function accesses array index values in EDR container.

Syntax

```
Long edrArrayIndex(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The array block of the EDR container whose index you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the index of the EDR container.

Example

```
edrArrayIndex( DETAIL.ASS_TCF_AAA_DETAIL.PCM_OP_TCF_AAA_AUTHORIZE.INPUT.PIN_FLD_
BALANCES, 0, 0, 0, 0) = 1;
edrIndex = edrArrayIndex( DETAIL.ASS_TCF_AAA_DETAIL.PCM_OP_TCF_AAA_
AUTHORIZE.OUTPUT.PIN_FLD_BALANCES, 0, 0, 0, 0);
```

edrClearErrors

This function clears the list of errors that the pipeline modules add to the EDR container.

Each pipeline module error has a name, severity level, and optional parameters that you can use for debugging or constructing an error message. The error list is a collection of all the errors that the pipeline modules have added to an EDR, the number of errors in the list, and the maximum severity of the errors. You can use the errors to reject an EDR or to instruct the pipeline module to process an EDR differently or to not process an EDR.

However, if an EDR does not have errors severe enough to be rejected or processed differently, you can use this function to remove the errors from the list. This function resets the error count to 0 and the maximum severity level to normal.

Important: Before clearing the errors, analyze all the errors in the EDR to ensure they can be safely ignored.

Syntax

```
Void edrClearErrors();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
function onInvalidDetailEdr
{
  if(edrNumErrors() > 0)
   {
   logStdout(" Current Edr contains" + longToStr(edrNumErrors()) +      "Errors");
   edrClearErrors();
   logStdout(" Current Edr contains" + longToStr(edrNumErrors()) +       "Errors
after clearErrors");
   }
  else
   {
   logStdout(" Current Edr contains no Errors");
   }
}
```

edrConnectToken

This function associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field.

This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.

Syntax

```
Bool edrConnectToken(EdrField field [, Long idx1 [, Long idx2 ...]], const String
tokenName);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

tokenName
The name of the token field to access (stream record field).

Return Values

Returns **true** if the EDR field is successfully associated with the input token. Returns **false** if the function fails.

Example

```
Bool success = edrConnectToken(DETAIL.RECORD_TYPE, "SOL42.DETAIL.RECORD_NUMBER");
```

edrConnectTokenEx

This function associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field.

This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.

Syntax

```
Bool edrConnectTokenEx(String field, Long indicesArray, Long numIndices, String
tokenName);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

tokenName
The name of the token field to access (stream record field).

Return Values

Returns **true** if the EDR field is successfully associated with the input token. Returns **false** if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.RECORD_TYPE";
numberOfIndices = 0;

Bool success = edrConnectTokenEx(edrFieldName, indicesArray, numberOfIndices,
"SOL42.DETAIL.RECORD_NUMBER");
```

edrContainsAdditionalStream

This function determines whether an EDR has an additional output stream with the name you pass in. EDRs contain one default stream and any number of additional output streams.

Syntax

```
Bool edrContainsAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the output stream you want to confirm exists in the EDR.

Return Values

Returns **true** if the stream exists. Returns **false** if it does not.

Example

```
if ( edrContainsAdditionalStream( "TELOut3" ) == false )
{
logStdout( "ERROR: EDR does not contain additonal stream: TELOut1\n" );
}
```

edrCurrentTokenIndex

This function returns the index of the token parsed from the stream. It is valid only in input grammar.

Syntax

```
Long edrCurrentTokenIndex();
```

Parameters

None.

Return Values

Returns the token index if the token exists. Returns **-1** if the function fails.

Example

```
Long index = edrCurrentTokenIndex();
logStdout("Currently processing: " + edrTokenString(index0 + "\n");
```

edrDate

This function retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values. When setting date values, use the function as the left-hand value in an assignment statement.

Syntax

```
Date edrDate(EdrField field [, Long idx1 [, Long idx2 ... ]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the date value of the EDR field if the function is successful. Returns **INVALID_DATE** if the data type for this EDR is not **Date** or if the path through the EDR tree structure is not valid.

Example

```
Date timeStamp;

timeStamp = edrDate( DETAIL.CHARGING_START_TIMESTAMP ); \
edrDate( DETAIL.CHARGING_START_TIMESTAMP ) = sysdate();
```

edrDateEx

This function retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values. When setting date values, use the function as the left-hand value in an assignment statement.

Syntax

```
Date edrDateEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices

Return Values

Returns the date value of the EDR field if the function is successful. Returns **INVALID_DATE** if the data type for this EDR is not **Date** or if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGING_START_TIMESTAMP";
numberOfIndices = 0;

Date timeStamp;

timeStamp = edrDateEx( edrFieldName, indicesArray, numberOfIndices); \
edrDateEx( edrField, indicesArray, numberOfIndices) = sysdate();
```

edrDecimal

This function retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values. When used to set decimal values, use the function as the left-hand value in an assignment statement.

Syntax

```
Decimal edrDecimal(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the decimal value of the EDR field if the function is successful. Returns an invalid decimal value if the data type for this EDR is not decimal or if the path through the EDR tree structure is not valid (for example, an index number is wrong).

Example

```
Decimal oldAmount;

oldAmount = edrDecimal( DETAIL.CHARGED_AMOUNT_VALUE ); \
edrDecimal( DETAIL.CHARGED_AMOUNT_VALUE ) = oldAmount + 1.0;
```

edrDecimalEx

This function retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values. When used to set decimal values, use the function as the left-hand value in an assignment statement.

Syntax

```
Decimal edrDecimalEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

level
Number of indices.

Return Values

Returns the decimal value of the EDR field if the function is successful. Returns an invalid decimal value if the data type for this EDR is not decimal or if the path through the EDR tree structure is not valid (for example, an index number is wrong).

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGED_AMOUNT_VALUE";
numberOfIndices = 0;

Decimal oldAmount;

oldAmount = edrDecimalEx(edrFieldName, indicesArray, numberOfIndices); \
edrDecimalEx(edrFieldName, indicesArray, numberOfIndices) = oldAmount + 1.0;
```

edrDelete

This function deletes the current EDR container, changing the current pointer to the EDR container directly in front of the deleted EDR.

Syntax

```
Bool edrDelete();
```

Parameters

None.

Return Values

Returns **true** if the current EDR container is deleted successfully. Returns **false** if there was no current EDR container.

Example

```
if ( edrDelete() )
{
  logStdout( "EDR container deleted" );
}
```

edrDeleteDatablock

This function deletes a data block from the current EDR container. The function is not supported for nested transactions (for example, transactions contained within transactions).

Syntax

```
Bool edrDeleteDatablock(EdrField block, Long idx1 [, Long idx2 ...]);
```

Parameters

block
The name of the data block you want to delete.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the data block is successfully deleted. Returns **false** if the operation fails.

Example

```
if edrDeleteDatablock( DETAIL.ASS_GSMW_EXT, 0 ) == false )
{
  logStdout("Error: failed to delete datablock");
}
```

edrDeleteField

This function clears the contents of a field in an EDR container. The function is not supported for nested transactions (for example, transactions contained within transactions).

Syntax

```
Bool edrDeleteField(EdrField field, Long idx1 [, Long idx2 ...]);
```

Parameters

field
The name of the EDR field you want to delete.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the EDR field content is successfully deleted. Returns **false** if the operation fails.

Example

```
if edrDeleteField( DETAIL.ASS_GSMW_EXT.RECORD_NUMBER ) == false )
{
  logStdout("ERROR: failed to delete field");
}
```

edrDuplicate

This function duplicates the current EDR container. The returned index is used as a parameter for the **edrSetCurrent** function to access the newly created EDR container.

Syntax

```
Long edrDuplicate();
```

Parameters

None.

Return Values

Returns the index of the duplicate EDR container (the index is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
Long index = edrDuplicate();
if ( index < 0 )
{
  logFormat( "ERROR: duplication of edr failed" );
}
else
{
  if ( edrSetCurrent( index ) == true )
  {
    // send new edr to duplicate output
    edrSetStream( "DuplicateOutput" );
  }
}
```

edrEmptyInput

This function sets the state of a field to **EDR_INPUT_EMPTY** when the field is present in the CDR but contains no value.

Syntax

```
Bool edrEmptyInput(EdrField field, Long idx1 [, Long idx2 ... ]);
```

Parameters

field
The name of the empty EDR field.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Bool success = edrEmptyInput(DETAIL.BASIC_SERVICE);
```

edrFieldConnectInfo

This function retrieves the Info string associated with the token for the corresponding EDR field. By default, the Info string contains the description of the token type. This is the default for ASCII object types.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
String edrFieldConnectInfo(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the Info string associated with the token for the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
logStdout("This field is of type: "+ edrFieldConnectInfo\
( DETAIL.RECORD_TYPE ) +"\n" );
```

edrFieldTokenBytePos

This function calculates the position of the token associated with the corresponding EDR field. The calculation is in bytes starting from the beginning of the input file.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
Long edrFieldTokenBytePos(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the position (in bytes) of the token associated with the EDR field if the function is successful. Returns **-1** if the EDR field is not associated with a token.

Example

```
if ( edrString( DETAIL.RECORD_TYPE ) != "020" )
{
  logStdout("Error, unexpected value at bytePosition= \
  "+ longToStr(edrFieldTokenBytePos( DETAIL.RECORD_TYPE )) + \
  "\n" );
}
```

edrGetAdditionalStream

This function gets the name of an additional EDR output stream given an array index number.

Each EDR contains a default output stream and any number of additional output streams.

Syntax

```
String edrGetAdditionalStream(Long index_number);
```

Parameter

index_number
The array index of the output stream that you need the name of.

Return Values

Returns the name of the stream if the function is successful. Returns an empty string for all other conditions.

Example

```
String streamName = edrGetAdditionalStream( 5)

if ( streamName == "" )
{
logStdout( "ERROR: no additional stream set at index: 5\n" );
}
```

edrGetError

This function retrieves the names of the attached error messages.

Syntax

```
String edrGetError(Long idx);
```

Parameter

idx
The index of the error to be retrieved.

Return Values

Returns the name of the attached error if the function is successful. Returns an empty string if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrGetErrorParameters

This function retrieves the parameters associated to a specified error.

Syntax

```
Long edrGetErrorParameters(Long idx, Array params);
```

Parameters

idx
The index of the error that you want to retrieve, where 0 <= *idx* < **edrNumErrors**.

params
The string array where the parameters can be stored. This is a return parameter.

Return Values

Returns the number of parameters in the array. Returns **0** if this function fails or if there are no parameters in the array.

Example

```
String paramList[];
Long paramCount;
Long Tap3MaxParamCount = 7;
long i;
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  if (edrGetError(i) == "ERR_TAP3_RET")
  {
    // get parameter list
    paramCount = edrGetErrorParameters(i, paramList);
    // check if enough parameters
    if (paramCount != Tap3MaxParamCount)
    {
      logStdout( "ERROR " + longToStr(i) + ": " + edrGetError(i) \
      + ", has missing parameters\n" );
    }
  }
}
```

edrGetErrorSeverity

This function retrieves the severity for each of the associated errors.

Syntax

```
Long edrGetErrorSeverity(Long idx);
```

Parameter

idx
The index of the error whose severity is being retrieved.

Return Values

Returns **0** if the severity of the attached error is Normal. Returns **1** if the severity of the attached error is Warning. Returns **2** if the severity of the attached error is Minor. Returns **3** if the severity of the attached error is Major. Returns **4** if the severity of the attached error is Critical. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + " Severity: " + \
  longToStr(edrGetErrorSeverity(i)) + "\n" );
}
```

edrGetStream

This function gets the output stream for an EDR.

Syntax

```
String edrGetStream();
```

Parameters

None.

Return Values

Returns the name of the actual string.

Example

```
String streamName = edrGetStream();
```

edrHasError

This function retrieves the names of the attached error messages.

Syntax

```
Bool edrHasError(String error);
```

Parameter

error
The name of the error to be retrieved.

Return Values

Returns the name of the attached error if the function is successful. Returns an empty string if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrInputState

This function retrieves the input state of an EDR field.

Syntax

```
Long edrInputState(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the EDR field for which to return the input state.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **1** if the EDR field contains a default value that was added due to missing input data in the CDR. Returns **2** if the EDR field contains a default value that was added due to empty input data in the CDR. Returns **3** if the EDR field is not populated or contains data that came from the CDR.

Example

```
Bool boolvar;
boolvar = edrEmptyInput(DETAIL.BASIC_SERVICE);
boolvar = edrMissingInput(DETAIL.QOS_USED);
switch(edrInputState(DETAIL.BASIC_SERVICE))
{
  case EDR_INPUT_MISSING:
    logStdout("DETAIL.BASIC_SERVICE: MISSING\n");
    break;
  case EDR_INPUT_EMPTY:
    logStdout("DETAIL.BASIC_SERVICE: EMPTY\n");
    break;
  default:  // "uninteresting" values
    logStdout("DETAIL.BASIC_SERVICE: OTHER\n");
    break;
}
```

edrInternalState

This function returns the internal state of an EDR field.

Syntax

```
Long edrInternalState(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the EDR field for which to return the internal state.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns 0 if cleared. Returns 1 if connected. Returns 2 if initialized. Returns 3 if set. Returns 4 if restored. Returns 5 if restored asset. Returns -1 if the function fails.

Example

```
Long state = edrInternalState(DETAIL.ASS_CBD.CP.CHARGE);
```

edrInternalStateEx

This function returns the internal state of an EDR field.

Syntax

```
Long edrInternalStateEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns 0 if cleared. Returns 1 if connected. Returns 2 if initialized. Returns 3 if set. Returns 4 if restored. Returns 5 if restored asset. Returns -1 if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD.CP.CHARGE";
indicesArray[0]=0;
indicesArray[1]=0;
numberOfIndices=2;

Long state = edrInternalStateEx(edrFieldName, indicesArray, numberOfIndices);
```

edrIsValidDetail

This function determines whether the current EDR container is a valid detail container. This helps you avoid processing of EDR containers that will be discarded.

Syntax

```
Bool edrIsValidDetail();
```

Parameter

None.

Return Values

Returns **true** if the current EDR container is a valid detail container. Returns **false** if it is not a valid detail container.

Example

```
if ( edrIsValidDetail() == true )
{
  // process the edr
}
```

edrLong

This function retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values. When setting Long values, use the function as left-hand value in an assignment statement.

Syntax

```
Long edrLong(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the Long value of the EDR field if the function is successful. Returns **0** if the EDR has no Long field or if the path through the EDR tree structure is not valid.

Example

```
edrLong( DETAIL.CHARGED_TAX_RATE ) = 1600;
```

edrLongEx

This function retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values. When setting Long values, use the function as left-hand value in an assignment statement.

Syntax

```
Long edrLongEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the Long value of the EDR field if the function is successful. Returns **0** if the EDR has no Long field or if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGED_TAX_RATE";
numberOfIndices=0;
edrLongEx(edrFieldName, indicesArray, numberOfIndices) = 1600;
```

edrMaxSeverity

This function finds the maximum severity of the errors added to the current EDR container.

Syntax

```
Long edrMaxSeverity();
```

Parameters

None.

Return Values

Returns the maximum severity of the errors of the EDR container if the function is successful. Returns **0** if there are no errors. Returns **-1** if the function fails.

Example

```
if ( edrMaxSeverity() == 0 )
{
  // The edr has no errors with severity > 0
}
```

edrMissingInput

This function sets the state of a field to **EDR_INPUT_MISSING** when the field is not present in the CDR.

Syntax

```
Bool edrMissingInput(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the missing EDR field.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Bool success = edrMissingInput(DETAIL.QOS_USED);
```

edrNumDatablocks

This function determines the number of data blocks of the specified type.

Syntax

```
Long edrNumDatablocks(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The name of the data block you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the number of data blocks (the number is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
for ( i = 0; i < edrNumDatablocks( DETAIL.ASS_CBD ); i = i + 1 )
{
  String recordType = edrString( DETAIL.ASS_CBD.RECORD_TYPE, i );
}
```

edrNumDatablocksEx

This function determines the number of data blocks of the specified type.

Syntax

```
Long edrNumDatablocksEx(String block, Long indicesArray, Long numIndices);
```

Parameters

block
The name of the data block you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the number of data blocks (the number is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD";
numberOfIndices=0;

for ( i = 0; i < edrNumDatablocksEx(edrFieldName, indicesArray, numberOfIndices);
i = i + 1 )
{
  String recordType = edrString( DETAIL.ASS_CBD.RECORD_TYPE, i );
}
```

edrNumErrors

This function accesses the number of error messages attached to the current EDR container.

Syntax

```
Long edrNumErrors();
```

Parameters

None.

Return Values

Returns the number of attached error messages (this number will be greater than or equal to 0) if the function is successful. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrNumTokens

This function accesses the number of tokens attached to the current EDR container.

Syntax

```
Long edrNumTokens();
```

Parameters

None.

Return Values

Returns the number of attached tokens (this number will be greater than or equal to 0) if the function is successful. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumTokens(); i = i+1 )
{
  logStdout( "Token " + longToStr(i) + ": " + \
  edrGetToken(i) + "\n" );
}
```

edrRemoveAdditionalStream

This function removes additional output streams from an EDR. Each EDR has a default output stream and any number of additional output streams.

Note: This function will not remove the default output stream.

Syntax

```
Bool edrRemoveAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the output stream that you are removing from the EDR.

Return Values

Returns **true** if the function is successful or if the named stream does not exist. Returns **false** for all other conditions.

Example

This example shows how to use **edrRemoveAdditionalStream** to remove an output stream.

```
if ( edrRemoveAdditionalStream( "TELOut1" ) == false
{
logStdout( "ERROR: failed to remove additional stream: TELOut1\n" );
}
```

Example 7–1 Example removeoutmod.isc file

This example removes output module streams:

```
removeoutmod.isc
---------------
function onDetailEdr
   {
      if (edrRemoveAdditionalStream( "TelOut1" ) == true)
   {
      logStdout("Stream TelOut1 removed ");
   }
      if (edrRemoveAdditionalStream( "TelOut2" ) == true)
      {
         logStdout("Stream TelOut2 removed ");
      }
   } // end onDetailEdr + end iScript
```

edrSetContentType

This function sets the content type of the current EDR container.

Syntax

```
Bool edrSetContentType(Long content);
```

Parameter

content
The content type to be assigned to the EDR container:

- EDR_UNKNOW_CONT
- EDR_HEADER
- EDR_DETAIL
- EDR_TRAILER
- EDR_START
- EDR_STOP
- EDR_BEGIN
- EDR_END
- EDR_BEGIN_TRANSACTION
- EDR_END_TRANSACTION

Return Values

Returns **true** if the content type is valid. Returns **false** if the container type is not valid.

Example

```
if ( edrSetContentType( EDR_TRAILER ) == false )
{
  logFormat( "ERROR: edrSetContentType() failed" );
}
```

edrSetCurrent

This function sets the current EDR container. All EDR container functions only access the current EDR container.

Syntax

```
Bool edrSetCurrent(Long index);
```

Parameter

index
The index of the EDR container you want to set. This is the return value from **edrDuplicate**.

Return Values

Returns **true** if there is an EDR container with the specified index. Returns **false** if there is no EDR container with that index.

Example

```
Long index = edrDuplicate();
if ( index < 0 )
{
  logFormat( "ERROR: duplication of edr failed" );
}
else
{
  // Set the output stream for the old container
  edrSetStream( "OrigOutput" );

  // Set the output stream for the new container
  if ( edrSetCurrent( index ) == true )
  {
    edrSetStream( "NewOutput" );
  }
}
```

edrSetIsValidDetail

This function sets the EDR container's valid detail flag. The valid detail flag specifies whether the EDR container is to be discarded.

Syntax

```
Void edrSetIsValidDetail(Bool flag);
```

Parameter

flag
The valid detail flag for the EDR container.

Return Values

Returns nothing.

Example

```
if ( ... )
{
  // record shall be discarded
  edrSetIsValidDetail( false );
}
```

edrSetStream

This function sets the output stream for an EDR. Internally, Pipeline Manager uses stream numbers instead of stream names. For this reason, the name specified must be converted to a number. If you use a constant as the stream name, the conversion can be performed at compile time, resulting in quicker performance than using a stream name that is not a constant. The second advantage of using a constant is that the existence of the stream can be checked at compile time.

Caution: Illegal stream names lead to compilation errors.

Syntax

```
Bool edrSetStream(String streamName);
```

Parameter

streamName
The name of the output stream for the EDR container.

Return Values

Returns **true** if the output stream is successfully set. Returns **false** if the output stream does not exist.

Example

```
// This is the FAST method: The stream number can be evaluated \
   at compile time.
// There is also a check if the stream exists at compile time.
if ( edrSetStream( "NationalOutput" ) == false )
{
  logFormat( "ERROR: edrSetStream() failed" );
}

// This is the SLOW method and should be avoided.
String nationalOutput = "NationalOutput"

if ( edrSetStream( nationalOutput ) == false )
{
  logFormat( "ERROR: no stream " + nationalOutput );
}
```

edrString

This function retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values. When setting string values, use this function as the left-hand value in an assignment statement.

Syntax

```
String edrString(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the string value of the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
if ( edrString( DETAIL.RECORD_TYPE) == "020" )
edrString(DETAIL.RECORD_TYPE) = "021";
```

edrStringEx

This function retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values. When setting string values, use this function as the left-hand value in an assignment statement.

Syntax

```
String edrStringEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the string value of the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.RECORD_TYPE";
numberOfIndices=0;

if ( edrStringEx(edrFieldName, indicesArray, numberOfIndices) == "020" )
edrStringEx(edrFieldName, indicesArray, numberOfIndices) = "021";
```

edrTokenString

This function retrieves the content of each token, as identified by their indexes. When the index is not available, as for a function call with no argument, this function returns the complete byte string attached to the EDR. The byte string corresponds to the original input string that generated the EDR.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
String edrTokenString([Long idx]);
```

Parameter

idx
The index of the token whose index you want to retrieve, where 0 <= *idx* < **edrNumTokens**.

Return Values

Returns the contents of the tokens if the function is successful. Returns an empty string if the index is invalid or there are no tokens associated with the EDR.

Example

```
logStdout( "The original (input) record corresponding to this \
EDR is \n" + edrTokenString() );
```

iRulesModeOn

This function enables the iRules mode. In the iRules mode, the init section does not consider the specified indices for an EDR field.

Syntax

```
iRulesModeOn();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
INIT_SCRIPT:
function testPrint
{
iRulesModeOff();
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,1));
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,2));
iRulesModeOn();
}
```

iRulesModeOff

This function disables the iRules mode. Disabling iRules mode ensures that the INIT takes the specified indices.

Syntax

```
iRulesModeOff();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
INIT_SCRIPT:
function testPrint
{
iRulesModeOff();
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,1));
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,2));
iRulesModeOn();
}
```

pipelineName

This function retrieves the name of the pipeline in which the script is running.

Syntax

```
String pipelineName();
```

Parameters

None.

Return Values

Returns the pipeline name.

Example

```
logPipeline("This script runs in pipeline " + pipelineName());
```

stopPipeline

This function stops the pipeline from which it is called. After the pipeline is stopped, the operator must restart the pipeline using the **ifw** command.

Note: This function does not work within the BEGIN function because the pipeline object instantiation is not completed when the BEGIN function is executed.

Important: Use this function only when there is an unrecoverable error that requires operation intervention.

Syntax

```
Void stopPipeline();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
if (unrecoverableError())
{
stopPipeline();
}
```

File Manipulation Functions

Table 7–7 contains file manipulation functions.

Table 7–7 File Manipulation Functions

Function	Description
fileClose	Closes a file that was opened earlier using the **fileOpen** function.
fileCopy	Copies a file.
fileDelete	Deletes a file.
fileEof	Checks to see whether the end of file has been reached.
fileFlush	Flushes the contents of the file buffer to disk.
fileIsOpen	Determines whether a file is currently open.
fileOpen	Opens a file for reading or writing. If the file is already open, the old file will be closed and the new file will be opened. The open mode is equivalent to the **fopen** C function.
fileReadLine	Reads a line from the input file. The line is read until the function encounters an end-of-line or end-of-file character or until *maxLen* is reached.
fileRename	Renames a file.
fileSeek	Sets the read/write pointer on a specific position (in bytes from the beginning of the file) in an opened file.
fileTell	Retrieves the position (measured in bytes from the start of the file) of the read/write pointer in an opened file.
fileWriteLong	Writes a Long value, as a string and not in binary mode, to the output file.
fileWriteStr	Writes a string to the output file.

fileClose

This function closes a file that was opened earlier using the **fileOpen** function.

Syntax

```
Void fileClose(File file);
```

Parameter

file
The file you want to close.

Return Values

Returns nothing.

Example

```
File out;
if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteStr( out, "Hello World!" );
  fileClose( out );
}
```

fileCopy

This function copies a file.

Syntax

```
Bool fileCopy(String old, String new);
```

Parameters

old
The file name of the file to be copied.

new
The file name of the copy.

Return Values

Returns **true** when a file has been copied. Returns **false** when it has not been copied.

Example

```
if ( fileCopy(tempName, realname ) == false )
{
logStdout( "Failed to copy" + tempName + " to " + realName );
}
```

fileDelete

This function deletes a file.

Syntax

```
Void fileDelete(String file);
```

Parameter

file
The name of the file you want to delete.

Return Values

Returns **true** if the file was successfully deleted. Returns **false** if the function failed.

Example

```
if ( fileDelete( "test.txt" ) == false )
{
  logFormat( "ERROR: failed to delete 'test.txt'" );
}
```

fileEof

This function checks to see whether the end of file has been reached.

Syntax

```
Bool fileEof(File file);
```

Parameter

file
The file you want to check.

Return Values

Returns **true** if the end of the file was reached or if no file was open. Returns **false** if it does not reach the end of the file.

Example

```
while ( fileReadLine( in, line, 2048 ) == true )
{
  ...
}
if ( fileEof( in ) == false )
{
  logFormat( "ERROR: read error()" );
}
```

fileFlush

This function flushes the contents of the file buffer to disk.

Syntax

```
Bool fileFlush(File file);
```

Parameter

file
The file you want to flush.

Return Values

Returns **true** if the file was successfully flushed. Returns **false** if the function failed.

Example

```
fileWriteStr( out, "Price is " + price );
if ( fileFlush( out ) == false )
{
  logFormat( "ERROR: fileFlush() failed" );
}
```

fileIsOpen

This function determines whether a file is currently open.

Syntax

```
Bool fileIsOpen(File file);
```

Parameter

file
The name of file you want to check.

Return Values

Returns **true** if the file is open. Returns **false** if the function failed.

Example

```
if ( fileIsOpen( in ) == false )
{
  logFormat( "ERROR: file is not open" );
}
```

fileOpen

This function opens a file for reading or writing. If the file is already open, the old file will be closed and the new file will be opened. The open mode is equivalent to the **fopen** C function.

Syntax

```
Bool fileOpen(File file, String fileName, String openMode);
```

Parameters

file
The file you want to open.

fileName
The name of the file you want to open.

openMode
The string specifying the open mode. Specify this parameter as you would for the **fopen** C function. The following description of open mode is from the Linux ManPage:

- **r:** Open text file for reading. The stream is positioned at the beginning of the file.
- **r+:** Open for reading and writing. The stream is positioned at the beginning of the file.
- **w:** Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
- **w+:** Open for reading and writing. The file is created if it does not exist; otherwise it is truncated. The stream is positioned at the beginning of the file.
- **a:** Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.
- **a+:** Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file.

Return Values

Returns **true** if the file was opened successfully. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == false )
{
  logFormat( "ERROR: fileOpen() failed" );
}
```

fileReadLine

This function reads a line from the input file. The line is read until the function encounters an end-of-line or end-of-file character or until *maxLen* is reached.

Syntax

```
Bool fileReadLine(File file, String line, Long maxLen);
```

Parameters

file
The name of file you want to read.

line
The string that specifies the line to be read. This must be a left-hand value.

maxLen
The maximum length for the line.

Return Values

Returns **true** if the line is successfully read. Returns **false** if the function failed.

Example

```
File in;
String line;

if ( fileOpen( in, "test.txt", "r" ) == true )
{
  fileReadLine( in, line, 100 );
}
```

fileRename

This function renames a file. The new name can specify a different directory, but both the old and new file must be in the same file system.

Syntax

```
Bool fileRename(String old, String new);
```

Parameters

old
The old file name.

new
The new file name.

Return Values

Returns **true** if the file is successfully renamed. Returns **false** if the function failed.

Example

```
if ( fileRename( tempName, realName ) == false )
{
  logStdout( "Failed to rename " + tempName + " to " + realName );
}
```

fileSeek

This function sets the read/write pointer on a specific position (in bytes from the beginning of the file) in an opened file.

Syntax

```
Bool fileSeek(File file, Long offset);
```

Parameters

file
The file in which you want to set a read/write pointer.

offset
The position where you want to set the read/write pointer.

Return Values

Returns **true** when setting the read/write pointer in an opened file is successful. Returns **false** when it has not been successful.

Example

```
long offset = fileTell( myfile );
if ( fileSeek(myfile, offset) == false )
{
logStdout( "could not set the file read/write pointer to " + longToStr(offset) );
}
```

fileTell

This function retrieves the position (measured in bytes from the start of the file) of the read/write pointer in an opened file.

Syntax

```
Long fileTell(File file);
```

Parameter

file
The file to check.

Return Values

Returns the position of the read/write pointer when successful. Returns **(-1)** when an error occurs.

Example

```
long offset = fileTell( Myfile );
if ( offset != (-1) )
{
logStdout( "the read pointer is currently on position " + longToStr() + " to " +
realName );
}
```

fileWriteLong

This function writes a Long value to the output file. The Long value is written as a string and not in binary mode.

Syntax

```
Bool fileWriteLong(File file, Long value [, Long len [, Bool leading [, String pad]]]);
```

Parameters

file
The file you want to write the Long value to.

value
The Long value to write.

len
The length of the output.

leading
Specifies whether to add leading or trailing characters: **true** adds leading characters, **false** adds trailing characters.

pad
The padding character to use as the first character of the string.

Return Values

Returns **true** if the Long value is successfully written. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteLong( out, 100, 14, true, "0" );
}
```

fileWriteStr

This function writes a string to the output file. The string is not automatically terminated by an end-of-line character.

Syntax

```
Bool fileWriteStr(File file, String string);
```

Parameters

file
The file you want to write the string to.

string
The string to write.

len
The length of the output. This parameter is optional.

leading
Specifies whether to add leading or trailing characters: **true** adds leading characters, **false** adds trailing characters.

pad
The padding character to use as the first character of the string.

Return Values

Returns **true** if the string is successfully written. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteStr( out, "Hello World!\n" );
}
```

Flist Manipulation Functions

Table 7–8 contains flist manipulation functions.

Table 7–8 Flist Manipulation Functions

Function	Description
fListToString	Returns the content of the current flist in string format.
fListFromString	Replaces the current flist with an flist based on the input string.
fListCount	Counts the number of elements at the top level of the current flist.
fListCreateNew	Replaces the current flist with an empty flist.
fListDate	Retrieves the date value from the current flist.
fListDecimal	Retrieves the decimal value from the current flist.
fListDropElem	Removes an array from the current flist.
fListDropFld	Deletes a field from the current flist.
fListElemid	Retrieves the array element ID from the specified array field.
fListGetErrorText	Puts the field name from the flist into *string1* and the error text into *string2*.
fListLong	Retrieves the long value from the current flist.
fListNumElem	Counts the number of elements in an array in the current flist.
fListPopElem	Resets the array to the previous value.
fListPushElem	Creates and sets the array element into which other functions set field values.
fListSetDate	Sets a date field in the current flist.
fListSetDecimal	Sets a decimal field in the current flist.
fListSetLong	Sets a long value within a PIN_FLDT_INT or PIN_FLDT_EMUN field in the current flist.
fListSetPoid	Sets a POID field in the current flist.
fListSetString	Sets a string field in the current flist.
fListString	Retrieves the string value from the current flist.
opcodeExecuteInternal	Calls the opcode specified in the parameter.

fListToString

This function returns the content of the current flist in string format. The function calls "PIN_FLIST_TO_STR".

Syntax

```
String fListToString();
```

Parameters

None.

Return Values

Returns the content of the current flist in string format. Returns an empty string on failure.

Example

```
logStdout(fListToString());
fListCreateNew();
```

fListFromString

This function removes the current flist and replaces it with an flist based on a string that you pass in as a parameter. The function calls PIN_STR_TO_FLIST.

Syntax

```
Bool fListFromString(const String flist_str);
```

Parameter

flist_str
The contents of the flist to be created, in string format.

Return Values

Returns **true** on success and **false** on failure.

Example

```
String flistStr =

"0    PIN_FLD_ARRAY          ARRAY [0] allocated 13, used 1" +
"1    PIN_FLD_STRING         STR [0] \"testing\"" +
"1    PIN_FLD_DECIMAL        DECIMAL [0] 0.000" +
"1    PIN_FLD_INT            INT [0] 60";

if(!fListFromString(flistStr))
{
// flist could not be parsed
}
```

fListCount

This function counts the number of elements at the top level of the current flist by calling PIN_FLIST_COUNT.

Syntax

```
Long fListCount();
```

Parameters

None.

Return Values

Returns the number of elements at the top level of the current flist. Returns **-1** on failure.

Example

```
Long resultCounts = fListCount();
```

fListCreateNew

This function removes the current flist and replaces it with an empty flist.

Syntax

```
Bool fListCreateNew();
```

Parameters

None.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListCreateNew();
```

fListDate

This function retrieves the date value from a PIN_FLDT_TSTAMP field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Date fListDate([const String path_field [, Long elem_id]] [,const String path_
field2 [, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the date is retrieved.

Return Values

Returns the date value from the specified PIN_FLDT_TSTAMP field. Returns INVALID_DATETIME on failure.

Example

```
fListDate("PIN_FLD_RESULTS",1,"PIN_FLD_CREATED_T");
```

fListDecimal

This function retrieves the decimal value from a PIN_FLDT_DECIMAL field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Decimal fListDecimal([const String path_field [, Long elem_id]] [,const String
path_field2 [, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the decimal value is retrieved.

Return Values

Returns the decimal value from the specified PIN_FLDT_DECIMAL field. Returns INVALID_DECIMAL on failure.

Example

```
fListDecimal("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_ORDER");
```

fListDropElem

This function removes an array from the current flist by calling PIN_FLIST_ELEM_DROP.

Syntax

```
Bool fListDropElem(const String array_field [,Long = 0 elem_id]);
```

Parameters

array_field
The name of the array.

elem_id
The array's element ID. The default is **0**.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListDropElem("PIN_FLD_ARGS", 2);
```

fListDropFld

This function deletes a field from the current flist by calling PIN_FLIST_FLD_DROP.

Syntax

```
Bool fListDropFld(const String field)
```

Parameter

field
The name of the field to be deleted.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListDropFld("PIN_FLD_LABEL");
```

fListElemid

This function retrieves the array element ID from the specified array field using a 0-n index in the array.

Syntax

```
Decimal fListElemid([const String path_field [, Long elemid]]
                    [,const String path_field2 [, Long elemid]
                     ... , ] const String array_field, Long index);
```

Parameters

path_field
A parent substruct or array field that is part of the path to the target array. The parameter is repeated in the case of nested arrays.

elem_id
The element ID of a parent array or substruct.

field
The name of the array from which the element ID is retrieved.

index
The 0-n index of the exact array element, the ID of which to return.

Return Values

Returns the *elem_id* value of the array element specified by 0-n index. Returns **INVALID_ARRAY** on failure.

Example

```
fListElemid("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 0);
```

fListGetErrorText

This function puts the field name from the flist into *string1* and the error text into *string2*. You can use the error information for logging or other purposes.

Syntax

```
Void fListGetErrorText(String string1, String string2);
```

Parameters

string1
String field into which the field name is placed.

string2
String field into which the error text is placed.

Return Values

Returns nothing.

Example

```
// Opcode failed
   String s1;
   String s2;
   fListGetErrorText(s1, s2);
```

fListLong

This function retrieves the long value from a PIN_FLDT_INT or PIN_FLDT_ENUM field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Long fListLong([const String path_field [, Long elem_id]] [,const String path_
field2 [, Long elem_id] ... , ]const String field) ;
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the long value is retrieved.

Return Values

Returns the long value from the specified PIN_FLDT_INT or PIN_FLDT_ENUM field. Returns 0 on error.

Example

```
fListLong("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_LENGTH")
```

fListNumElem

This function counts the number of elements in a PIN_FLD_ARRAY field by calling PIN_FLIST_ELEM_COUNT. If the array is stored in substructs or other arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Long fListNumElem([const String path_field [, Long elem_id]] [,const String path_
field2 [, Long elem_id] ... , ] const String array_field, Long elem_id);
```

Parameters

path_field
A substruct or array field that is part of the path to the target array. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

array_field
The name of the array.

Return Values

Returns the number of elements in the specified array. Returns **-1** on failure.

Example

```
Long resultCounts = fListNumElem("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6);
```

fListPopElem

This function resets the array to the previous value.

Syntax

```
Void fListPopElem();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
fListPopElem();
```

fListPushElem

This function creates and sets the array element into which other functions set field values. The function calls PIN_FLIST_ELEM_ADD.

Syntax

```
Bool fListPushElem(const String array_field [,Long = 0 element]);
```

Parameters

array_field
The name of the array to set.

element
The array's element ID. The default is **0**.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListPushElem("PIN_FLD_ARGS", 2);
```

fListSetDate

This function sets a date field in the current flist.

Syntax

```
Bool fListSetDate(const String field, Date value);
```

Parameters

field
The name of the date field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Date d = strToDate("20060402143600"); // Apr 2, 2006 2:36 pm
fListSetDate("PIN_FLD_EFFECTIVE_T", d);
```

fListSetDecimal

This function sets a decimal field in the current flist.

Syntax

```
Bool fListSetDecimal(const String field, Decimal value);
```

Parameters

field
The name of the decimal field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetDecimal("PIN_FLD_DECIMAL",edrDecimal(DETAIL.ASS_DATA.VALUE,1));
```

fListSetLong

This function sets a long value in a PIN_FLDT_INT or PIN_FLDT_ENUM field in the current flist.

Syntax

```
Bool fListSetLong(const String field, Long value);
```

Parameters

field
The name of the long field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetLong("PIN_FLD_INT",edrLong(DETAIL.ASS_DATA.QUANTITY, 1));
```

fListSetPoid

This function sets a POID field in the current flist.

Syntax

```
Bool fListSetPoid(String field, String poid);
```

Parameters

field
The name of the POID field to set.

poid
The POID string to be set in the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Bool success = fListSetPoid( "PIN_FLD_POID", "0.0.0.1 /account 1099832 0" );
```

fListSetString

This function sets a string field in the current flist.

Syntax

```
Bool fListSetString(const String field, String value);
```

Parameters

field
The name of the string field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetString("PIN_FLD_USAGE_TYPE", usageClass);
```

fListString

This function retrieves the string value from a PIN_FLDT_STR or PIN_FLDT_POID field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
String fListString([const String path_field [, Long elem_id]] [,const String path_
field2 [, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the string value is retrieved.

Return Values

Returns the string value from the specified PIN_FLDT_STR or PIN_FLDT_POID field. Returns NULL_STRING on failure.

Example

```
fListString("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_DESCR")
```

opcodeExecuteInternal

This function calls the opcode specified in the parameter. You can call any opcode.

You use this function in iScripts that run in a real-time pipeline. The function uses the Connection Manager (CM) context information in the EDR to call the opcode through the existing connection.

See "opcodeExecute" for information about calling opcodes in batch pipelines.

Before calling **opcodeExecuteInternal**, you compose the input flist by using the flist extension functions. The input flist is stored and used internally by the opcode call.

The output flist of the opcode call is also stored internally and replaces the input flist. It can be retrieved by using the flist extension functions again.

If there is an error in the opcode call, an error buffer will be set. The error text can be retrieved with the **fListGetErrorText**function. The error text can then be logged.

Syntax

```
Bool opcodeExecuteInternal(Long opcode, Long flags);
```

Parameters

opcode
The opcode number of the opcode to be executed.

flags
The opcode flag value. Flag values differ from opcode to opcode. Some opcodes do not expect a flag value. Use **0** for opcodes that do not expect a flag value.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Long PCM_OP_SEARCH = 7;
...
if ( opcodeExecuteInternal(PCM_OP_SEARCH, 0) == false )
....
```

Hash and Array Functions

Table 7–9 contains hash and array functions.

Table 7–9 Hash and Array Functions

Function	Description
arrayClear	Clears an array.
arraySize	Determines the size of an array.
hashClear	Clears a hash.
hashContains	Checks to determine whether a hash-array contains a specific value.
hashKeys	Retrieves all keys used in an associative array.
hashRemove	Removes an entry from an associative array.

arrayClear

This function clears an array.

Syntax

```
Void arrayClear(Array array);
```

Parameter

array
The array you want to clear.

Return Values

Returns nothing.

Example

```
if ( arraySize( array ) > 0 )
{
  // Cleanup the array
  arrayClear( array );
}
```

arraySize

This function determines the size of an array.

Syntax

```
Long arraySize(Array array);
```

Parameter

array
The array whose size you want to determine.

Return Values

Returns the size of the array.

Example

```
for ( i = 0; i < arraySize( array ); i = i + 1 )
{
  logStdout( "array[" + longToStr(i) + "] = " + array[i] );
}
```

hashClear

This function clears a hash.

Syntax

```
Void hashClear(Hash hash);
```

Parameter

hash
The hash you want to clear.

Return Values

Returns nothing.

Example

```
// Cleanup the hash
hashClear( hash );
```

hashContains

This function checks to determine whether a hash-array contains a specific value.

Syntax

```
Void hashContains(Hash hash, String key);
```

Parameters

hash
The hash you want to search.

key
The value you want to search for.

Return Values

Returns **true** if the hash contains the value specified by *key*. Returns **false** if the hash does not contain this value.

Example

```
if ( hashContains( hash, "Hamburg" ) == true )
{
  logStdout( "The hash contains a value for 'Hamburg'" );
}
```

hashKeys

This function retrieves all keys used in an associative array.

Syntax

```
Long hashKeys(Hash hash, Array key);
```

Parameters

hash
The hash you want to search, looking for the key.

key
The string array as a return buffer for the keys.

Return Values

Returns the number of elements in the hash.

Example

```
String keys[];
Long age{};
Long i;

age{"Mary"} = 23;
age{"John"} = 18;

Long entries = hashKeys( age, keys );
for ( i = 0; i < entries; i = i+1 )
{
  logStdout( "Age of " + keys[i] + " is " + \
  longToStr( age{keys[i]} ) + "\n" );
}
```

hashRemove

This function removes an entry from an associative array.

Syntax

```
Bool hashRemove(Hash hash, String key);
```

Parameters

hash
The hash from which you want to remove the entry.

key
The entry to remove.

Return Values

Returns **true** if the element was removed successfully. Returns **false** if the function failed.

Example

```
if ( hashRemove( hash, "Hamburg" ) == true )
{
  logStdout( "The entry 'Hamburg' was removed from the hash\n" );
}
```

Mapping Functions

Table 7–10 contains mapping functions.

Table 7–10 Mapping Functions

Function	Description
longDecode	Maps Long values to other Long values.
strDecode	Maps string values to other string values.

longDecode

This function maps Long values to other Long values.

Syntax

```
Long longDecode(Long toMap, Long default [[, const Long src1, const Long dest1]
...]);
```

Parameters

toMap
The Long value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = longDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

strDecode

This function maps string values to other string values.

Syntax

```
String strDecode(String toMap, String default [[, const String src1, const String
dest1] ...]);
```

Parameters

toMap
The string value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = strDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

Opcode Calling Functions

Table 7–11 contains opcode calling functions.

Table 7–11 Opcode Calling Functions

Function	Description
opcodeExecute	Calls the specified opcode.
opcodeGetConnection	Obtains a connection from the specified connection pool.
pcmOpCatch	Calls PCM_OP, which performs the operation of the specified opcode and then returns the contents of the error buffer (**ebuf**) produced by the operation.

opcodeExecute

This function calls the opcode specified in the parameter. You can call any opcode.

You use this function to call opcodes in batch pipelines. See "opcodeExecuteInternal" for information about calling opcodes from real-time pipelines.

Before calling **opcodeExecute** the first time in an iScript, you must call **opcodeGetConnection** to get the connection from the connection pool. If the CM restarts or if the existing connection is broken, an error results. To get a new connection, add more conditional checks for **opcodeExecute** and then call **opcodeGetConnection**.

For example:

```
.....
Bool connectionOpened;
Long PCM_OP_NUMBER = 200;
function onBeginEdr
{
  connectionOpened = false;
}
function getCMConnection
{
 if (connectionOpened == false)
 {
 {String connectionPool = "ifw.DataPool.CMConnectionPool.Module";
  connectionOpened = opcodeGetConnection(connectionPool);
 }
}
function Bool callOpcode
{
   Long retryCount;
   Bool success;
   Long numberOfRetries = 10;
   String fldName;
   String errMsg;

getCMConnection();
success = opcodeExecute(PCM_OP_NUMBER, 0);
if (success == false)
{
   fListGetErrorText (fldName, errMsg);
   if (errMsg == "PIN_ERR_CONNECTION_LOST")
   {
    connectionOpened = false;
    for (retryCount = 0; ((retryCount < numberOfRetries) and (connectionOpened ==
false)); retryCount = retryCount + 1)
    {
        connectionOpened = false
        getCMConnection();
        if(connectionOpened == true)
        {
           success = opcodeExecute(PCM_OP_NUMBER,0);
         }
    if ((connectionOpened == false) and (retryCount >= numberOfRetries))
    {
     logStdout("Error executing opcode PCM_OP_GET_PIN_VIRTUAL_TIME due to lost
connection with CM\n");
    }
```

```
     if ((success == false)
      {
        logStdout("Error: "+ errMsg + "while executing opcode PCM_OP_GET_PIN_
VIRTUAL_TIME\n");
       }
      return success;
    function  onDetailEdr()
    {
      Bool success = callOpcode()
    }
.....
```

Before calling **opcodeExecute**, you compose the input flist by using the flist extension functions. The input flist is stored and used internally by the opcode call.

The output flist of the opcode call is also stored internally and replaces the input flist. It can be retrieved by using the flist extension functions again.

If there is an error in the opcode call, an error buffer will be set. The error text can be retrieved with the **fListGetErrorText** function. The error text can then be logged.

Syntax

```
Bool opcodeExecute(Long opcode, Long flags);
```

Parameters

opcode
The opcode number of the opcode to be executed.

flags
The opcode flag value. Flag values differ from opcode to opcode. Some opcodes do not expect a flag value. Use **0** for opcodes that do not expect a flag value.

Return Values

Returns **true** on success and **false** on failure.

Example

```
...
Long PCM_OP_SEARCH = 7;
Bool success = opcodeExecute(PCM_OP_SEARCH, 0)
...
```

opcodeGetConnection

This function obtains a connection to the CM from the specified connection pool in a batch pipeline. You must configure a connection pool in the pipeline before using this function. See *DAT_ConnectionPool* in the BRM documentation for information about configuring a connection pool.

In an iScript, you must call **opcodeGetConnection** before calling **opcodeExecute** the first time. You do not need to call **opcodeGetConnection** again for subsequent opcode calls in the same script. Adding more conditional checks ensures that **opcodeGetConnection** is not called every time a CDR is processed.

> **Note:** This function is required in iScripts used in batch pipelines only. It is not necessary in real-time pipelines.

For example:

```
........
Bool connectionOpened;
function onBeginEdr
{
connectionOpened = false;
}
function getCMConnection
{
  if(connectionOpened == false)
   {
   String connectionPool = "ifw.DataPool.CMConnectionPool.Module";
   connectionOpened = opcodeGetConnection(connectionPool);
   }
  if(connectionOpened == false)
   {
      logStdout("Unable to get connection to CM\n");
   }
}
.........
```

Syntax

```
Bool opcodeGetConnection(String connectionPool);
```

Parameter

connectionPool
The full registry name of the connection pool used for the pipeline.

Return Values

Returns **true** on success and **false** on failure.

Example

```
...
String connectionPool = "ifw.DataPool.CMConnectionPool.Module";
Bool success = opcodeGetConnection(connectionPool);
...
```

pcmOpCatch

This function calls the PCM_OP opcode, which performs the operation of the specified opcode and then returns the contents of the error buffer (**ebuf**) produced by the operation. This enables the calling iScript to resolve any errors that occur during the operation without exiting the logic the iScript is performing.

For more information about the PCM_OP opcode, see "PCM_OP".

Syntax

```
pcmOpCatch(opcode, flags, in_flistp, ebufp);
```

Parameters

opcode
The name or number of the opcode whose operation PCM_OP is to perform. See "Base Opcodes" for possible opcodes.

> **Note:** Opcode numbers are listed in the **pcm_ops.h** file in the *BRM_Home*/**include** directory, where *BRM_Home* is the directory in which you installed the BRM components.

flags
The flags supported by the opcode being called. See the opcode description for information on the flags it takes.

- If the opcode takes no flags, enter **0**.
- To specify multiple flags, separate the flag names with a vertical bar (|).

in_flistp
A pointer to the input flist of the opcode being called. See the individual opcode flist reference pages for the input flist specifications.

ebufp
A pointer to the error buffer that stores any errors that occur during the operation.

Return Values

This function returns nothing.

Errors are passed back to the calling iScript through the specified error buffer.

Example

```
pcmOpCatch(7, SRCH_DISTINCT, search, PINERR);
```

Pipeline System Functions

Table 7–12 contains Pipeline system functions.

Table 7–12 Pipeline System Functions

Function	Description
formatName	Determines the name of the format the script is running in.
logFormat	Writes messages to the pipeline log
logPipeline	Writes messages to the pipeline log.
msgArg	Deprecated.
msgName	Deprecated.
msgNumArgs	Deprecated.
registryNodeName	Returns the name of the registry node in which the script (iScript or input/output grammar) is running.
regString	Retrieves values from the registry.
reqSend	Sends a request to a registered object and waits for an answer (i.e., synchronous messaging).
scriptUsable	Sets the *usable* flag for the script. If the *usable* flag is set to **false** in the BEGIN function during Pipeline Manager startup, Pipeline Manager will not start to process CDRs. The **false** setting can be useful if the iScript initialization fails.
sendEvent	Sends an event to the event handler.
stopFormat	Stops the format; for example, after critical errors.

formatName

This function determines the name of the format the script is running in.

Syntax

```
String formatName();
```

Parameters

None.

Return Values

Returns the format name.

Example

```
logFormat( "This script runs in format " + formatName() );
```

logFormat

This function writes messages to the pipeline log.

Important: This function is obsolete and should be replaced by the **logPipeline** function.

Syntax

```
Void logFormat(String msg);
```

Parameter

msg
The message to write to the pipeline log.

Return Values

Returns nothing.

Example

```
logFormat( "Hello World!" );
```

logPipeline

This function writes messages to the pipeline log.

Syntax

```
Void logPipeline(String msg [, Long severity]);
```

Parameters

msg
The message to write to the pipeline log.

severity
The severity of the message:

- **0** = Debug
- **1** = Normal
- **2** = Warning
- **3** = Minor error
- **4** = Major error
- **5** = Critical error

  The default is **0**.

Return Values

Returns nothing.

Example

```
logPipeline( "ERROR: critical database error occurred", 4 );
```

registryNodeName

This function returns the name of the registry node in which the script (iScript or input/output grammar) is running.

Syntax

```
String registryNodeName();
```

Parameters

None.

Return Values

Returns the name of the registry node in which the script (iScript or input/output grammar) is running.

Example

```
logFormat( "This script is located at registry: " + registryNodeName () );
//this will return the following result,
//This script is located at registry:
ifw.Pipelines.ciber25.Functions.Thread1.FunctionPool.myIScript.Module.Scripts.retr
ieve
```

regString

This function retrieves values from the registry.

Syntax

```
String regString(String name);
```

Parameter

name
The name of the registry entry.

Return Values

Returns the specified registry entry if it exists. Returns an empty string if there is no registry entry with that name.

Example

```
if ( regString( "IntegRate.DataPool.Customer.Module.Source" ) ==\
"FILE" )
{
  logFormat( "Customers are read from file" );
}
```

reqSend

This function sends a request to a registered object and waits for an answer (i.e., synchronous messaging).

Syntax

```
Bool reqSend(String reqDestination, String reqName, Array inParams, Array
outParams);
```

Parameters

reqDestination
The registry name of the request's destination.

reqName
The name of the request.

inParams
A string array containing the input parameter expected by the destination to be able to process the request.

outParams
A string array to contain the reply to the request.

Request Names

REQ_NEWSEQUENCENUMBER
(Sequencer) Returns the new sequence number.

REQ_CC
(Pipeline) Returns the country code defined in the registry for this pipeline.

REQ_MCC
(Pipeline) Returns the mobile country code defined in the registry for this pipeline.

REQ_NAC
(Pipeline) Returns the national access code value defined in the registry for this pipeline.

REQ_IAC
(Pipeline) Returns the international access code defined in the registry for this pipeline.

REQ_IAC_SIGN
(Pipeline) Returns the international access code sign value defined in the registry for this pipeline.

REQ_NDC
(Pipeline) Returns the national destination code value defined in the registry for this pipeline.

REQ_REJECT_STREAM_NAME
(Pipeline) Returns the reject stream name defined in the registry for this pipeline.

REQ_REJECT_STREAM
(Pipeline) Returns the reject stream number defined in the registry for this pipeline.

REQ_EVENTHANDLER_NAME
(ifw) Returns the event handler name.

REQ_ERROR_FILENAME
(Input) Returns the name and path of the error file.

REQ_INPUT_FILENAME
(Input) Returns the name and path of the input file.

REQ_INPUT_TEMP_FILENAME
(Input) Returns the name and path of the temporary input file.

REQ_DONE_FILENAME
(Input) Returns the name and path of the done file.

REQ_RETURN_FILENAME
(Input) Returns the name and path of the return file.

REQ_OUTPUT_FILENAME
(Output) Returns the name and path of the output file.

REQ_OUTPUT_TEMP_FILENAME
(Output) Returns the name and path of the temporary output file.

Return Values

Returns **true** if the request has been sent and an answer received successfully. Returns **false** if sending the request has failed.

Example

```
sendArray [0] = "abcdefg.so142" ;
if ( reqSend( reg_InputStream, "REQ_ERROR_FILENAME",sendArray, receiveArray)==true
)
{
String errFileName = receiveArray[0]; // the fully qualified filename (including
path)
}
```

scriptUsable

This function sets the *usable* flag for the script. If the *usable* flag is set to **false** in the BEGIN function during Pipeline Manager startup, Pipeline Manager will not start to process CDRs. The **false** setting can be useful if the iScript initialization fails.

> **Important:** You can use this function only in the iScript modules and not in the input grammar.

Syntax

```
Void scriptUsable(Bool usable);
```

Parameter

usable
The flag indicating whether the script is usable.

Return Values

Returns nothing.

Example

```
function BEGIN
{
  ...
  if ( fileOpen( inFile, "data.txt", "r" ) == false )
  {
    logFormat( "failed to open data file 'data.txt'" );
    scriptUsable( false );
  }
}
```

sendEvent

This function sends an event to the event handler.

Syntax

```
Bool sendEvent(String event [, String arg1 [, String arg2 ...]]);
```

Parameters

event
The name of the event to send.

argX
A comma-delimited number of argument strings used as parameters for the event.

Return Values

Returns **true** if the event was successfully sent. Returns **false** if the function failed.

Example

```
if ( sendEvent( EVT_FILE_PROCESSED, filename ) == false )
{
  logFormat( "ERROR: sendEvent() failed" );
};
```

stopFormat

This function stops the format; for example, after critical errors.

Syntax

```
Void stopFormat();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
if ( fileWriteString( out, data ) == false )
{
  logFormat( "ERROR: fileWriteString() failed" );
  stopFormat();
};
```

Static Functions

This section describes static functions.

EXT_ConvertCli::convert

This function normalizes wireless and wireline CLIs into international format.

Syntax

```
const BAS_String EXT_ConvertCli::convert(const BAS_String& cli,
                                         const BAS_String& modInd,
                                         const long typeOfNumber,
                                         const BAS_String& natAccessCode,
                                         const BAS_String& intAccessCode,
                                         const BAS_String& countryCode,
                                         const BAS_String& intAccessCodeSign,
                                         const BAS_String& natDestinCode );
```

Parameters

cli
CLI to normalize.

modInd
Modification Indicator, for example, "00".

typeOfNumber
Type Of Number, for example, 0.

natAccessCode
National Access Code, for example, "0".

intAccessCode
International Access Code, for example, "00".

countryCode
Country Code, for example, "49".

intAccessCodeSign
International Access Code Sign, for example, "+".

natDestinCode
National Destination Code, for example, "172".

Return Values

Returns a CLI in international normalized format: <iac>< cc><ndc>extension.

Example

```
...
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normCli;
BAS_String cli = "01721234567";

normCli = EXT_ConvertCli::convert( cli, "00", 0, "0", "00", "49", "+", "172" );

// normCli now contains: 00491721234567
...
```

EXT_ConvertIPv4::convert

This function normalizes IPv4 addresses.

Syntax

```
const BAS_String EXT_ConvertIPv4::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 3 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "192.168.1.253";

normIp = EXT_ConvertIPv4::convert( ip );

// normIp now contains: 192168001253

...
```

EXT_ConvertIPv6::convert

This function normalizes IPv6 addresses.

Syntax

```
const BAS_String EXT_ConvertIPv6::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "0:0:0:AF:E:0:1:FE";

normIp = EXT_ConvertIPv6::convert( ip );

// normIp now contains: 00000000000000AF000E0000000100FE

...
```

EXT_ConvertIPv4onv6::convert

This function normalizes IPv4 over IPv6 addresses. The decimal IPv4 address is converted into hexadecimal representation.

Syntax

```
const BAS_String EXT_ConvertIPv4onv6::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IPv6 address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "0:0:0:0:0:0:192.168.10.1";

normIp = EXT_ConvertIPv4onv6::convert( ip );

// normIp now contains: 00000000000000000000000C0A80A01

...
```

Standard Functions

Table 7–13 contains standard functions.

Table 7–13 Standard Functions

Function	Description
closeClientConnection	Closes the connection to the Diameter client
currentTimeInMillis	Gets the current system time in milliseconds.
getClientState	Gets the state of a Diameter client.
mutexAcquire	Acquires the mutex specified by the handle (a number that identifies the mutex). When the mutex specified by the handle is already acquired by another thread, the function call is blocked unless the other thread releases the mutex by calling the **mutexRelease** function.
mutexCreate	Creates a mutex that can later be accessed by its handle.
mutexDestroy	Used to destroy a mutex that is no longer needed.
mutexRelease	Releases a mutex that has been acquired. It unblocks a functional call by another thread that has been trying to acquire the mutex using the **mutexAcquire** function.
sleep	Makes the process sleep.
startTimer	Starts the timer.
sysExecute	Executes a command line in a file.
sysGetEnv	Gets an environment variable.

closeClientConnection

This function closes the connection to the Diameter client.

Syntax

Void closeClientConnection(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns nothing.

Example

```
if( (commandCode == DIA_DP_REQUEST) and (commandFlag == 0) )
{
  logPipeline("CommandCode: DIA_DP_REQUEST. Closing the connection.",0);
  closeClientConnection(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_
INFO.SOCKETID,0,0));
}
```

currentTimeInMillis

This function gets the current system time in milliseconds.

You can use this function in your custom iScript to record the time when a pipeline or a pipeline module starts processing an EDR and when it finishes processing the EDR. You can then calculate the difference between the start and end times to determine the latency of the EDR processing in a pipeline or module.

You can include the iScript at any point in a pipeline to determine the latency of an EDR processing between two points in a pipeline.

Syntax

```
Long currentTimeInMillis();
```

Parameters

None.

Return Values

Returns the current system time as a long value.

Example

This example gets the current system time and logs a message:

```
logStdout("The Time in milliseconds is = " + longToStr(currentTimeInMillis()) +
"\n");
```

getClientState

This function gets the state of a Diameter client.

Syntax

Long getClientState(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns one of the following state values:

- 0 = STATE_INITIAL
- 1 = STATE_OKAY
- 2 = STATE_DOWN

Example

state = getClientState(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.SOCKETID,0,0));

mutexAcquire

This function acquires the mutex specified by the handle (a number that identifies the mutex). When the mutex specified by the handle is already acquired by another thread, the function call is blocked unless the other thread releases the mutex by calling the **mutexRelease** function.

Syntax

```
Bool mutexAcquire(Long handle);
```

Parameter

handle
The handle of the mutex to acquire.

Return Values

Returns **true** if a valid handle is used and the mutex is acquired. Returns **false** if an invalid handle is used and the mutex is not acquired.

Example

```
// enter the protected area
mutexAcquire (handle)

// protected area

//leave the protected area
mutexRelease(handle)
```

mutexCreate

This function creates a mutex that can later be accessed by its handle.

Syntax

```
Long mutexCreate();
```

Parameters

None.

Return Values

Returns a handle (**>0**) if the mutex was created successfully. Returns **<0** if the mutex was not created successfully.

Example

```
long handle; function BEGIN
{
handle = mutexCreate ( )
if (handle < 0)
{
 logStdout("Mutex creation failed\n");
}
}
```

mutexDestroy

This function destroys a mutex that is no longer needed.

Syntax

```
Bool mutexDestroy(Long handle);
```

Parameter

handle
The handle of the mutex to be destroyed.

Return Values

Returns **true** when destroying the mutex is successful. Returns **false** when destroying the mutex has not been successful.

Example

```
if ( mutexDestroy (handle) == false )
{
logStdout( "Illegal mutex handle\n");
}
```

mutexRelease

This function releases a mutex that has been acquired. It unblocks a functional call by another thread that has been trying to acquire the mutex using the **mutexAcquire** function.

Syntax

```
Bool mutexRelease(Long handle);
```

Parameter

handle
The handle of the mutex you want to release.

Return Values

Returns **true** when a valid handle was used and the mutex is released successfully. Returns **false** when the handle used is invalid and the mutex is not released.

Example

```
// enter the protected area
mutexAcquire (handle)

// protected area

// leave the protected area
mutexRelease(handle)
```

sleep

This function makes the process sleep.

Syntax

```
Void sleep(Long seconds);
```

Parameter

seconds
The number of seconds you want the process to sleep.

Return Values

Returns nothing.

Example

```
sleep (10)
```

startTimer

This function starts the timer.

Syntax

Void startTimer(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns nothing.

Example

startTimer(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_
INFO.SOCKETID,0,0));

sysExecute

This function executes a command line in a file. When you call this function in an iScript, you must configure an EventHandler in the pipeline registry file. For example:

```
EventHandler
{
  ModuleName = EVT
  Module
  {
   Events
    {
    }
   Buffer
    {
     Size = 1000
    }
  }
}
```

See "Event Handler" in *BRM Configuring Pipeline Rating and Discounting*.

Syntax

```
Long sysExecute(String commandLine [String returnBuffer, Long timeToWait]);
```

Parameters

commandLine
The command line to execute. The value must be the path to an executable, followed by any arguments.

returnBuffer
A string to collect the output produced on stdout by *commandLine*. The stdin and stderr for *commandLine* will be the terminal.

timeToWait
The maximum time (in seconds) to wait for the response from the event handler. Command execution is terminated when *timeToWait* expires.

Return Values

Returns a Long value greater than 0 if the function is successful. Returns **-1** if the specified path points to a file that is either not readable or not executable.

Example

```
// list the contents of the /data/input directory

String cmdline = "/usr/bin/ls -l /data/input";
String retbuf;
Long timeToWait = 10; // 10 seconds
Long retval = sysExecute( cmdline, retbuf, timeToWait );
if ( retval != -1 )
{
  // code to process retbuf
  logStdout( retbuf );
}
```

sysGetEnv

This function specifies an environment variable you want returned.

Syntax

```
String sysGetEnv(String envVariable);
```

Parameter

envVariable
The name of the environment variable you want returned.

Return Values

Returns the specified environment variable and its settings.

Example

```
logStdout("******************-\n");
logStdout( "PATH=" + sysGetEnv( "PATH") +"\n");directory \n");
```

String Functions

Table 7–14 contains string functions.

Table 7–14 String Functions

Function	Description
decimalToStr	Converts a decimal value into a string.
decimalToStrHex	Converts a decimal value into a hexadecimal string. Use **round(***value***)** or **trunc(***value***)** to remove the decimal portion if you do not want it to be coded in hexadecimal.
longToHexStr	Converts a Long value into a hexadecimal string.
longToStr	Converts a Long value into a string.
strByteValue	Converts the first character in the input string to its byte value.
strDecode	Maps string values to other string values.
strEndsWith	Checks to see if a string ends with a special suffix.
strHexStrToStr	Converts each pair of characters in a given hexadecimal string into the equivalent single-byte ASCII character in a new string. The returned string is half the size of the original. Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets will cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.
strHexToDecimal	Converts a hexadecimal string to a decimal value.
strHexToLong	Converts a hexadecimal string into a Long value.
strLength	Determines the length of a string.
strMatch	Compares a regular expression to a string, looking for a match.
strPad	Pads a string to a specific length. The padding character and the justification can be selected.
strReplace	Replaces substrings in a string.
strSearch	Searches for a substring inside another string.
strSearchRegExpr	Searches for a regular expression to a string.
strSplit	Splits a string according to a specific separator character and stores the resulting tokens in a string array.
strStartsWith	Checks to see if a string starts with a specified prefix.
strStrip	Removes special leading or trailing characters from a string.
strStrToHexStr	Converts each character in a given string into its two-character hexadecimal equivalent in a new string. The returned string is twice the size of the original. Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.
strSubstr	Extracts a substring from a string.
strToDate	Converts a string into a date value.
strToDecimal	Converts string values to decimal values.

Table 7–14 (Cont.) String Functions

Function	Description
strToLong	Converts a string value to a Long value.
strToLower	Converts a string to lowercase characters.
strToUpper	Converts a string to uppercase characters.

decimalToStr

This function converts a decimal value into a string.

Syntax

```
String decimalToStr(Decimal value [, Long precision]);
```

Parameters

value
The value to convert into a string.

precision
The number of digits after the decimal point.

Return Values

Returns the value as a string.

Example

```
logFormat( "Pi = " + decimalToStr(pi) );
logFormat( "Pi = " + decimalToStr(pi,2) );
```

decimalToStrHex

This function converts a decimal value into a hexadecimal string.

> **Note:** Use **round(***value***)** or **trunc(***value***)** to remove the decimal portion if you do not want it to be coded in hexadecimal. For example, use **round(0)** to omit the .000 if you want only integer values returned.

Syntax

```
String decimalToStrHex(Decimal value [, String separator [, Long precision]]);
```

Parameters

value
The decimal value to convert into a hexadecimal string. Code this in readable ASCII.

separator
The character you want to use as a decimal separator (the default is **.**).

precision
The precision of the decimal value to use when generating the hexadecimal string (the default is **0**).

Return Values

Returns the decimal value as a hexadecimal string.

Example

```
logFormat( "X = " + decimalToStr(x) + "(" + decimalToStrHex(x) + \
" hexadecimal)" );
```

longToHexStr

This function converts a Long value into a hexadecimal string.

Syntax

```
String longToHexStr(Long value);
```

Parameter

value
The Long value to convert into a hexadecimal string.

Return Values

Returns the value as a hexadecimal string.

Example

```
logFormat( "X = " + longToStr(x) + "(" + longToHexStr(x) + \
" hexadecimal)" );
```

longToStr

This function converts a Long value into a string.

Syntax

```
String longToStr(Long value);
```

Parameter

value
The Long value to convert into a string.

Return Values

Returns the value as a string.

Example

```
logFormat( "X = " + longToStr(x) );
```

strByteValue

This function converts the first character in the input string to its byte value.

Syntax

```
Long strByteValue(String string);
```

Parameter

string
The string whose first character you want to convert.

Return Values

Returns the byte value of the first character if the function is successful. Returns **0** if the string is empty.

Example

```
Long ascA = strByteValue( "A" );
logStdout( "ASCII(A) = " + longToStr( ascA ) + "\n" );
```

strDecode

This function maps string values to other string values.

Syntax

```
String strDecode(String toMap, String default [[, const String src1, const String
dest1] ...]);
```

Parameters

toMap
The string value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = strDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

strEndsWith

This function checks to see if a string ends with a special suffix.

Syntax

```
Bool strEndsWith(String string, String suffix);
```

Parameters

string
The string to check the suffix for.

suffix
The suffix to check.

Return Values

Returns **true** if the string ends with the specified suffix. Returns **false** if the string does not end with the suffix.

Example

```
if ( strEndsWith( filename, ".txt" ) )
{
  logFormat( "file suffix is .txt" );
}
```

strHexStrToStr

This function converts each pair of characters in a given hexadecimal string into the equivalent single-byte ASCII character in a new string. The returned string is half the size of the original. For example, if you pass the string **58595A373839** to **strHexStrToStr**, it returns the string **XYZ789**.

Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets will cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.

Syntax

```
String strHexStrToStr(source);
```

Parameter

source
The hexadecimal string to convert to ASCII:

- It must have an even number of characters.
- Only numeric characters and A through F are permitted.
- It cannot be empty.

Return Values

Returns the string converted to ASCII if the function is successful.

If *source* has hexadecimal representations for embedded nulls, the returned string contains embedded nulls. The caller must interpret such strings correctly.

Example

```
String source = ""58595A373839";
String result = strHexStrToStr(source);
logStdout(result);
```

strHexToDecimal

This function converts a hexadecimal string to a decimal value.

Syntax

```
Decimal strHexToDecimal(String string [, String separator [, Long precision]]);
```

Parameters

string
The hexadecimal string (coded in readable ASCII) to convert into a decimal value.

separator
The character you want to use as decimal separator (the default is **.**).

precision
The precision of the decimal value to be generated (the default is **0**).

Return Values

Returns a decimal value when the value entered for *string* is successfully converted to a decimal value. Returns **0.0** if *string* is not a valid hexadecimal decimal/Long value and is therefore not converted to a decimal value.

Example

```
logStdout ( "1FF hex is " + decimalToStr ( strHexToDecimal ( "1FF" ) ) + "
decimal\n" );
```

strHexToLong

This function converts a hexadecimal string into a Long value.

Syntax

```
Long strHexToLong(String string);
```

Parameter

string
The hexadecimal string to convert into a Long value.

Return Values

Returns the hexadecimal string as a Long value.

Example

```
logStdout( "1FF hex is " + strHexToLong( "1FF" ) + " decimal\n" );
```

strLength

This function determines the length of a string.

Syntax

```
Long strLength(String string);
```

Parameter

string
The string whose length you want to determine.

Return Values

Returns the string length in characters if the function is successful.

Example

```
if ( strLength( edrString( DETAIL.RECORD_TYPE ) ) != 3 )
{
  logFormat( "WARNING: illegal RECORD_TYPE" );
};
```

strMatch

This function compares a regular expression to a string, looking for a match.

Syntax

```
String strMatch(String string, String regExp [, Long index]);
```

Parameters

string
The string that you want to search for the regular expression.

regExp
The regular expression to match against the string.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the matching part of the string if the function is successful. Returns **0** if the function does not find a match.

Example

```
if ( strMatch( filename, ".*\\.edr" ) != "" ) // IMPORTANT: the first \ is removed
by the compiler!!!!
{
  logFormat( filename + " is a *.edr file" );
}
```

strPad

This function pads a string to a specific length. The padding character and the justification can be selected.

> **Note:** The original string you started with will be truncated. If the original string is greater in length than the string you set up to result from applying the **String strPad** function.

Syntax

```
String strPad(String string, String padChar, Long length, Bool isLeftJustified);
```

Parameters

string
The string to pad (or truncate) to a specified length.

padChar
The pad character to use (the first of the string is used if empty).

length
The desired length of the returned string. If *length* is less than or equal to 0, an empty string is returned.

isLeftJustified
If set to **true**, it specifies that the string be left justified. If set to **false**, it specifies that the string be right justified.

Return Values

Returns the padded or truncated string.

Example

```
String resString;
resString = strPad ("hello", " ", 2, true); // -> resString = "he";
resString = strPad ("hello", " ", 2, false); // -> resString = "he";
resString = strPad ("hello", " ", 10, true); // -> resString = "hello ";
resString = strPad ("hello", " ", 10, false); // -> resString = "  hello";
resString = strPad ("hello", "0", 10, false); // -> resString = "00000hello";
resString = strPad ("hello", " ", -2, true); // -> resString = "";
```

strReplace

This function replaces substrings in a string.

Syntax

```
String strReplace(String toReplace, Long pos, Long len, String replace);
```

Parameters

toReplace
The string in which you want the substring replaced.

> **Important:** The input string in *toReplace* is not changed.

pos
The start position of the substring to replace. Positions start with 0.

len
The length of the substring to replace.

replace
The replacement string.

Return Values

Returns a string with the replacement string in the correct position. Returns an empty string if *pos* and *len* do not specify a valid substring.

Example

```
logFormat( strReplace( "Hello !", 5, 1, "World " ) );
```

strSearch

This function searches for a substring inside another string.

Syntax

```
Long strSearch(String string, String search [, Long index]);
```

Parameters

string
The string that you want to search.

search
The string that you want to search for.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the starting index (this should be a value greater than or equal to 0) for the search within the string. Returns a value less than 0 if the function does not find the string.

Example

```
if ( strSearch( edrString( DETAIL.B_NUMBER ), "0049", 0 ) >= 0 )
{
  logFormat( "B-Number contains '0049'" );
}
```

strSearchRegExpr

This function searches for a regular expression to a string.

Syntax

```
Long strSearchRegExpr(String string, const String regExp [, Long index]);
```

Parameters

string
The string that you want to search.

regExp
The regular expression to look for in the string.

> **Important:** The strSearchRegExpr function does not support braces ({ }); for example, A{1,3}.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the position index (this should be a value greater than or equal to 0) of the string if the function is successful. Returns a value less than 0 if the function does not find the string.

Example

```
if ( strSearchRegExpr( filename, ".*\\.doc", 0 ) >= 0 )
// IMPORTANT: the first \ is removed by the compiler
{
  logFormat( filename + " is a *.doc file" );
}
```

strSplit

This function splits a string according to a specific separator character and stores the resulting tokens in a string array.

Syntax

```
Long strSplit(Array res, String string, String sep);
```

Parameters

res
The resulting array to fill.

string
The input string to split.

sep
The separator to use for splitting. If the separator you specify is longer than one character, the function uses only the first character.

Return Values

Returns the number of elements in the resulting array.

Example

```
String ListArray[];
String ListString;
ListArray="first,second,third"
Long nbElem = strSplit( ListArray, ListString, "," );
for (Long i=0 ; i<nbElem ; i=i+1)
{
  logStdout( "Element " + ListArray[i] + "\n");
}
```

strStartsWith

This function checks to see if a string starts with a specified prefix.

Syntax

```
Bool strStartsWith(String string, String prefix);
```

Parameters

string
The string in which to check for the specified prefix.

prefix
The specified prefix being checked for in the string.

Return Values

Returns **true** if the string starts with the specified prefix. Returns **false** if the string does not start with the specified prefix.

Example

```
if ( strStartsWith( edrString( DETAIL.B_NUMBER ), "0049" ))
{
isNationalCall = true;
}
```

strStrip

This function removes special leading or trailing characters from a string.

Syntax

```
Bool strStrip(String string [, Long stripMode [, String stripChar]]);
```

Parameters

string
The string from which you want to remove leading or trailing characters.

stripMode
The strip mode:

- STRIP_LEADING
- STRIP_TRAILING
- STRIP_BOTH

The default is **STRIP_LEADING**.

stripChar
The character to be removed, which is the first or last character of the string (the default is the space character).

Return Values

Returns the stripped string.

Example

```
String test = "--------Hello-------------";
if ( strStrip( test, STRIP_BOTH, "-" ) == "Hello" )
{
  logStdout( "strStrip() works correct" );
}
```

strStrToHexStr

This function converts each character in a given string into its two-character hexadecimal equivalent in a new string. The returned string is twice the size of the original. For example, if you pass the string **XYZ789** to **strStrToHexStr**, it returns the string **58595A373839**.

Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.

Syntax

```
String strStrToHexStr(source);
```

Parameter

source
The ASCII string to convert to hexadecimal. It cannot be empty. Embedded nulls are permitted and handled correctly.

Return Values

Returns the string converted to hexadecimal if the function is successful.

Example

```
String source = "XYZ789";
String result = strStrToHexStr(source);
logStdout(result);
```

strSubstr

This function extracts a substring from a string.

Syntax

```
String strSubstr(String string, Long pos, Long len);
```

Parameters

string
The string from which you want to extract the substring.

pos
The start position of the substring to extract. Positions start with 0.

len
The length of the substring to extract.

Return Values

Returns the specified string if the function is successful. Returns an empty string if *pos* and *len* do not specify a valid substring.

Example

```
if ( strLength( string ) > 6 )
{
  string = strSubstr( string, 0, 6 );
}
```

strToDate

This function converts a string into a date value. The only supported string format is YYYYMMDDHHMMSS.

Syntax

```
Date strToDate(String dateStr);
```

Parameters

%%
The literal % character.

%d
The day of the month; for example, 29. The range is 00-31.

%H
The hour of the 24-hour day; for example, 14. The range is 00-23.

%m
The month of the year, from 01; for example, 02. The range is 01-12.

%M
The minutes after the hour; for example, 34. The range is 00-59.

%S
The seconds after the minute; for example, 56. The range is 00-59.

%y
The year of the century, from 00; for example, 04 for 2004. The range is 01-99. In most cases, you should avoid this parameter.

%Y
The year including the century; for example, 1994.

Return Values

Returns a valid date if the input string is in the right format. Returns an invalid date if the format is not correct.

Example

```
edrDate(DETAIL.CHARGING_START_TIMESTAMP) = \
strToDate("24.12.2002", "%d. %m. %Y");
```

strToDecimal

This function converts string values to decimal values.

Syntax

```
Decimal strToDecimal(String string);
```

Parameter

string
The string to convert to a decimal value.

Return Values

Returns the string converted to a decimal value if the function is successful. Returns **0** if the string is not a valid decimal value.

Example

```
x = x + strToDecimal( "13.32" );
```

strToLong

This function converts a numeric string value to a Long value. An alphanumeric string is returned as **0**.

Syntax

```
Long strToLong(String string);
```

Parameter

string
The string to convert to a Long value.

Return Values

Returns the string converted to a Long value if the function is successful. Returns **0** if the string is not a valid Long value.

Example

```
if ( strToLong( edrString(DETAIL.RECORD_TYPE) ) == 20 )
{
  // Basic detail record
}
```

strToLower

This function converts a string to lowercase characters.

Syntax

```
String strToLower(String string);
```

Parameter

string
The string to convert to lowercase characters.

Return Values

Returns the string converted to lowercase characters if the function is successful.

Example

```
if ( strToLower( "HELLO" ) == "hello" )
{
  ...
}
```

strToUpper

This function converts a string to uppercase characters.

Syntax

```
String strToUpper(String string);
```

Parameter

string
The string to convert to uppercase characters.

Return Values

Returns the string converted to uppercase characters if the function is successful.

Example

```
if ( strToUpper( "Hello" ) == "HELLO" )
{
  ...
}
```

Transaction Management Functions

Table 7–15 contains transaction management functions.

Table 7–15 Transaction Management Functions

Function	Description
edrDemandCancel	Sends a request to the Transaction Manager to cancel the current transaction.
edrDemandRollback	Sends a request to the Transaction Manager to roll back the current transaction.
edrRollbackReason	Allows the iScript module to request the reason for the rollback in the onRollback function.
tamItemType	Returns the type of an item in the currently processed transaction.
tamNumTransItems	Returns the number of items processed in the currently processed transaction.
tamStreamExtension	Used to access the extension value of each item in the current transaction.
tamStreamName	Used to access the stream name of each item in the current transaction.
tamTransId	Returns the transaction ID of the transaction currently being processed.

edrDemandCancel

This function sends a request to the Transaction Manager to cancel the current transaction.

Syntax

```
Bool edrDemandCancel();
```

Parameters

None.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrDemandCancel() == false )
{
  logStdout( "ERROR: failed to demand cancel" );
}
```

edrDemandRollback

This function sends a request to the Transaction Manager to roll back the current transaction.

Syntax

```
Bool edrDemandRollback([rollbackReason]);
```

Parameter

rollbackReason
The reason for the rollback.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

Request for rollback success status:

```
if ( edrDemandRollback() == false )
{
  logStdout( "ERROR: failed to demand rollback" );
}
```

Request for rollback with a reason:

```
edrDemandRollback("Invalid Input file")
```

edrRollbackReason

This function allows the iScript module to request the reason for the rollback in the onRollback function.

Syntax

```
String edrRollbackReason();
```

Parameters

None.

Return Values

Returns a string indicating the reason for the rollback.

Example

```
function Bool onRollback
{
  rollbackReason = edrRollbackReason();
  logStdout( "rollback reason= " + rollbackReason + "\n");
  return true;
}
```

tamItemType

This function returns the type of an item in the currently processed transaction. These items are only accessible for the functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Long tamItemType(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the type of the specified item:

- TAM_NORMAL
- TAM_RECYCLE
- TAM_RECYCLE_TEST

Returns a value of <0 if there is no current transaction in all other functions or the index is out of range.

Example

```
function onCancel
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    if ( tamItemType(i) == TAM_NORMAL )
    {
      ...
    }
  }
}
```

tamNumTransItems

This function returns the number of items processed in the currently processed transaction. The count includes only items accessible for the functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Long tamNumTransItems();
```

Parameters

None.

Return Values

Returns the number of items in the transaction currently being processed. Returns **0** if there is no current transaction in all other functions or there are no items in the current transaction.

Example

```
function onCancel
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    ...
  }
}
```

tamStreamExtension

This function accesses the extension value of each item in the current transaction. The index should be between 0 and **tamNumTransItems()–1**. Usually, the extension value contains the sequence number of the currently processed stream.

Syntax

```
String tamStreamExtension(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the stream extension string if the function is successful. Returns an empty string if the function fails.

Example

```
function onCommit
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    logFormat( "commiting " + tamStreamName(i) + \
    " with extension " + tamStreamExtension(i) );
  }
}
```

tamStreamName

This function accesses the stream name of each item in the current transaction. The index should be between 0 and **tamNumTransItems( )–1**.

Syntax

```
String tamStreamName(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the stream name if the function is successful. Returns an empty string if the function fails.

Example

```
function onCommit
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    logFormat( "committing " + tamStreamName(i) );
  }
}
```

tamTransId

This function returns the transaction ID of the transaction currently being processed. This function should only be used with functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Decimal tamTransId();
```

Parameters

None.

Return Values

Returns the current transaction ID. Returns **0.0** if there is no current transaction in the other functions.

Example

```
function onCancel
{
  Decimal transId = tamTransId();
  ...
}
```

8

Sample Applications

This chapter describes the sample programs included with the Oracle Communications Billing and Revenue Management (BRM) SDK, how to use the sample code, and how to run the sample programs.

Caution: These programs can change or delete data in your BRM database.

About Using the PCM C Sample Programs

BRM SDK includes a set of sample applications and templates using the Portal Communication Model (PCM) C application programming interface (API). You can use these sample programs and templates in the following ways:

- Use the sample programs as code samples for extending BRM components and applications and for writing custom applications.
- Run the corresponding executable application with a sample program to observe the changes it makes in BRM.
- Use the templates, which provide the basic structure for the components, to create your custom components, such as Facilities Modules (FMs) and Data Managers (DMs).

These samples are supported on several platforms: Linux, AIX, Solaris, and HP-UX IA64. Compile these sample programs using the appropriate compiler for your platform.

Finding the PCM C Sample Programs

You can view the sample programs by clicking the links to the sample programs. When you install BRM SDK on UNIX, sample programs and templates are found in the following directories:

- Most sample programs and the templates are installed in *BRM_SDK_Home*/**source/samples** by default.
- Other sample programs can be found in *BRM_SDK_Home*/**source/samples/apps/c**.
- Templates are located in *BRM_SDK_Home*/**source/templates**.

For information on installing BRM SDK on UNIX, see "Installing BRM SDK" in *BRM Installation Guide*.

Description of the PCM C Sample Programs

The sample programs demonstrate how to write code for various tasks when customizing BRM.

Each sample includes these supporting files:

- Source files to view or modify for your own applications.
- Makefiles to compile the sample programs on UNIX, if you make changes to the samples.
- A compiled application that verifies that the sample programs work as expected and that allows you to observe the changes the programs make in BRM.
- A **pin.conf** that allows you to specify the information required for the sample application to connect to BRM.

The following tables provide:

- A list of the sample programs and templates.
- A description of each sample program and template.
- Information on any executable program that you can run to observe the results.

Table 8–1 lists a sample for setting makefile macros.

Table 8–1 Setting Makefile Macros (File Located in BRM_SDK_Home/source/samples)

Sample	Description
env.unix	Shows you how the environment is set up, for example, the location of include directories. The makefiles reference the appropriate environment file for this information. Instructions on setting the makefile macros are included in these text files.

Table 8–2 lists the sample flist files.

Table 8–2 Creating an Flist (Files Located in BRM_SDK_Home/source/samples/flists/C)

Sample	Description
simple_flist.c	Shows how to create an flist with simple fields. Run **simple_flist.exe** to see a printout of the flist created, which contains a POID and two strings containing the first and last names. For information on how to run **simple_flist**, see "Running the Sample PCM C Programs".
flists_with_arrays.c	Shows how to create flists with arrays containing a single element and multiple elements. Run **flists_with_arrays.exe** to see the flists created by this sample. For information on how to run **flists_with_arrays**, see "Running the Sample PCM C Programs".
flists_with_substructs.c	Shows how to create an flist with a substructure. Run **flists_with_substructs.exe** to see the flists created by this sample. For information on how to run **flists_with_substructs**, see "Running the Sample PCM C Programs".

Table 8–3 lists a sample file for creating a context.

Table 8–3 Creating a Context (File Located in BRM_SDK_Home/source/samples/context/C)

Sample	Description
create_context.c	Shows you how to open a context, connect to BRM, perform operations, close the context and test if the connection is open. Run **CreateContext.exe** to see how to open a context. For information on how to run **create_context**, see "Running the Sample PCM C Programs".

Table 8–4 lists a sample file for calling an opcode.

Table 8–4 Calling an Opcode (File Located in BRM_SDK_Home/source/samples/callopcode/C)

Sample	Description
test_loopback.c	Shows you how to call an opcode. This sample calls the PCM_OP_TEST_LOOPBACK opcode which just returns the flist that you pass in as the input. Run **test_loopback.exe** to verify that the program returns input flist as the output. For information on how to run **test_loopback**, see "Running the Sample PCM C Programs".

Table 8–5 lists the sample files for client application functions.

Table 8–5 Creating a Client Application (Files Located in BRM_SDK_Home/source/samples/apps/c)

Sample	Description
sample_act.c	Shows how to generate activity for a service. For more information about this program, see "Creating Events by Using the sample_act.c Program".
sample_app.c	Shows how to create a customer account with services. For more information about this program, see "Creating Accounts by Using the sample_app.c Program".
sample_del.c	Shows how to remove accounts from BRM. For more information about this program, see "Removing Accounts by Using the sample_del.c Program".
sample_search.c	Shows how to search for objects and fields. For more information about this program, see "Searching by Using the sample_search.c Program".
sample_who.c	Shows how to display the current users. For more information about this program, see "Displaying Current Users by Using the sample_who.c Program".

Table 8–6 lists the FM template files.

Table 8–6 Templates for Creating an FM

Sample	Description
fm_generic_opcode.c	Provides structure for generic (FM) opcodes. See "Using the FM and DM Templates". This file is in *BRM_SDK_Home*/**templates/fm_template**.
fm_generic_config.c	Shows you how to map from the opcode to the function. See "Using the FM and DM Templates". This file is in *BRM_SDK_Home*/**templates/fm_template**.
op_define.h	Header file required by FM templates which defines **PCM_OP_GENERIC**. This file is in *BRM_SDK_Home*/**templates/fm_template**.

Table 8–7 lists the template file for creating a DM.

Table 8–7 Template for Creating a DM

Sample	Description
dm_generic.c	Shows the basic structure of a Data Manager. See "Using the FM and DM Templates". This file is in *BRM_SDK_Home*/**templates/dm_template**.

Table 8–8 lists the sample files for using the multithreaded application (MTA) APIs.

Table 8–8 Files for Using the Multithreaded Application (MTA) API

Sample	Description
pin_mta_monitor.c (located in *BRM_SDK_Home*/**bin**)	Sample monitoring utility.
pin_mta_test.c (located in *BRM_SDK_Home*/**source/samples/apps/c/mta_sample**)	Sample test program using the MTA framework.

Compiling the Sample PCM C Programs

In addition to using the sample programs as a working programming example, you can also use them as a basis for your own applications. You can make changes to the sample programs, compile, and run them to test your changes. The sample programs directory includes the following files:

- **env.unix** to set the environment
- Makefiles for UNIX to compile the samples

To compile the sample programs on UNIX:

1. Go to *BRM_SDK_Home*/**source/samples** and open **env.nt** or **env.unix**, depending on your operating system.
2. Set up the path for the environment by following the instructions in the file.
3. Save the file.
4. Compile using the appropriate **make** utility:

```
make
```

Running the Sample PCM C Programs

The executable versions of the sample programs are provided in addition to the source files. To see the output generated by a sample program, follow these basic steps:

1. Go to the directory where the sample program is located. The default structure is: *BRM_SDK_Home*/**source/samples** or *BRM_SDK_Home*/**source/samples/apps/c**.
2. Edit the entry in the configuration file **pin.conf** to point to the Connection Manager (CM).
3. Run the program by running the executable file, for example:

```
create_context.exe
```

Note: Some sample programs require parameters or have special syntax requirements. For more information, see "Creating Events by Using the sample_act.c Program", "Creating Accounts by Using the sample_app.c Program", "Removing Accounts by Using the sample_del.c Program", or "Searching by Using the sample_search.c Program".

Using the FM and DM Templates

In addition to the sample programs, the BRM SDK includes FM and DM templates that you can use as starting points for your own customized versions. You can make changes to the templates, compile them, and run them to test your changes. Makefiles and .dlls are provided for the templates in *BRM_SDK_Home*/**source/templates/fm_template** and *BRM_SDK_Home*/**source/templates/dm_template**.

The templates are provided in two forms:

- C files that you can modify and compile according to the instructions in Compiling the Sample PCM C Programs.
- DSP files that you can open as projects in Microsoft Visual Studio.

See "Testing New or Customized Policy FMs" and "Testing New or Customized DMs" in *BRM Developer's Guide* for information about testing the modified templates.

Creating Events by Using the sample_act.c Program

The **sample_act.c** program simulates customer activity by creating a session event for a service object. Use this program to generate any number of sessions to test new BRM functionality or custom opcodes.

For information on the structure and parameters, see the source file **sample_act.c** located in *BRM_SDK_Home*/**source/samples/apps/c**.

Syntax for sample_act.c

Run the program with appropriate parameters listed in Table 8–9 to specify the events you want to simulate. The options can be in any order.

Syntax for creating a typical IP telephony call

```
% sample_act [-c event_subtype] [-d duration_in_seconds] [-e session]
[-l login] [-s service_type] [-f] [-v] [-A phone_num_origin]
[-D phone_num_destination]
```

Syntax for creating a typical event

```
% sample_act [-c event_subtype] [-d duration_in_seconds] [-e session]
```

```
[-l login] [-s service_type] [-f] [-v]
```

Table 8–9 sample_act.c Execution Parameters

Parameter	Description	Condition
-b	Start time	Required
-c	Event subtype	Required
-d	Duration in seconds	Required
-e	Event type	Required
-f	Flist debugging on	Optional
-g	Lineage	Optional
-h	Help - starts usage program	Optional
-i	Impact category	Optional
-l	Login	Required
-n	RUM name	Optional
-q	Rate quantity	Required
-r	Rate name	Optional
-s	Service type	Required
-t	Test mode on	Optional
-v	Verbose status on	Optional
-?	Help - starts usage program	Optional

This example generates an IP session event:

```
% sample_act -v -e session -l login -d 3600 -s /service/ip
```

Creating Accounts by Using the sample_app.c Program

The **sample_app.c** program creates a new account with services in the specified plan. You can modify this program to add new services to an account or to create dummy accounts to test BRM functionality.

This program performs the following actions:

1. Opens a database channel
2. Retrieves the specified plan
3. Adds the customer information to the plan
4. Creates the customer account
5. Closes the database channel

For information on the structure and parameters, see the source file **sample_app.c** located in *BRM_SDK_Home*/**source/samples/apps/c**.

Syntax for sample_app.c

Run the program with appropriate options listed in Table 8–10, and plan name. The options can be in any order except that the name of the plan must be the last entry.

```
% sample_app  [-l login] [-p password] <plan>
```

Table 8–10 sample_act.c Account Creation Parameters

Parameter	Description	Condition
-l	Login	Required
-p	Password	Required
-d	Set error level	Optional
-h	Print standard error	Optional

The following example accepts the account logon and password for **jsmith**.

```
sample_app -l jsmith -p my_password email_plan
```

Removing Accounts by Using the sample_del.c Program

The **sample_del.c** program finds an account by searching for one of its service logins, and then deletes the account and all of its related objects.

> **Caution:** This program deletes accounts permanently. You cannot retrieve any accounts that you delete by running this program.

For information on the structure and parameters, see the source file, **sample_del.c** located in *BRM_SDK_Home***/source/samples/apps/c**.

Syntax for sample_del.c

The **sample_del.c** program does not take any parameters.

```
% sample_del  /servicetype login
```

This example deletes the **/service/ip** account with the login **smith**:

```
% sample_del /service/ip smith
```

Searching by Using the sample_search.c Program

The **sample_search.c** program demonstrates the different types of searches in BRM.

- Read-object search with single result expected

  Searches for the master account object and displays the results with PIN_FLIST_PRINT.

- Read-fields search with multiple results expected

  Searches for the POID, merchant, and status of all nonbillable accounts in the database.

- Step search

  Searches for services that require AES-encrypted passwords. The first 10 such services are retrieved in 2 blocks of 5 services each.

For information on the structure, see the source file **sample_search.c** located in *BRM_SDK_Home***/source/samples/apps/c**.

Syntax for sample_search.c

The **sample_search.c** program does not take any parameters.

```
% sample_search
```

Displaying Current Users by Using the sample_who.c Program

The **sample_who.c** program finds all the active dialup sessions in the database, looks up the login for each user with an open session, and displays a list of all customers currently logged in to your Internet service.

For information on the structure, see the source file **sample_who.c** located in *BRM_SDK_Home***/source/samples/apps/c**.

Syntax for sample_who.c

The **sample_who.c** program does not take any parameters.

```
% sample_who
```

Troubleshooting the sample_app.c Application

If you cannot run the **sample_app** application, use this information to identify any problems and resolve them.

Problem: Test Failed

```
sample# sample_app
bad/no "userid" from pin.conf file

Test Failed, See Log File.
```

Solution

Edit the **sample_app** configuration file to include the correct **userid** entry and make sure the application is configured correctly.

Problem: Bad Port Number

```
sample# sample_app
(11400): bad receive of login response, err 4
(11400): login failed 4

Test Failed, See Log File

sample# cat default.pinlog
E Fri Mar 15 14:56:44 1998  db2.corp <no name>:11393  pcm.c(1.41):90
         Connect open failed (4/100) in pcm_context_open
E Fri Mar 15 14:58:39 1998  db2.corp  <no name>:11400  pcm.c(1.41):90
         Connect open failed (4/5) in pcm_context_open
```

Solution

Edit the **cm_ptr** entry in the **sample_app** configuration file with the valid CM port number.

Problem: Customer Account Creation Error

```
sample# sample_app

Test Failed, See Log File

E Fri Mar 15 15.10:37 1998  db2.corp  :11405  sample_app.c:167
         op_cust_create_acct error [location= class= errno= field num= recid=<0>
```

```
reserved=<0>]
```

Solution

Load the BRM objects into the database.

About Using the PCM C++ Sample Programs

BRM SDK includes a set of sample applications using the PCM C++ API. You can use these sample programs in the following ways:

- Use the sample programs as code samples for extending BRM components and applications and for writing custom applications.
- Run the corresponding executable application with a sample program to observe the changes it makes in BRM.

These samples are supported on several platforms: Linux, AIX, Solaris, and HP-UX IA64. Compile these sample programs using the appropriate compiler for your platform.

Finding the Sample PCM C++ Programs

When you install BRM SDK on UNIX, the sample programs are installed by default in *BRM_Home*/**InfranetSDK/source/samples**.

For information on installing BRM SDK, see "Installing BRM SDK" in *BRM Installation Guide*.

You can also display the sample programs by clicking the links in this document.

Note: The installation directory is called *BRM_SDK_home* in the documentation.

Description of the Sample PCM C++ Programs

The sample programs demonstrate how to write code for various tasks when customizing BRM.

Each sample includes these supporting files:

- Source files to view or modify for your own applications
- Makefiles to compile the sample programs on UNIX, if you make changes to the samples
- A compiled application that verifies that the sample programs work as expected and that allows you to observe the changes the programs make in BRM
- A configuration file **pin.conf** that allows you to specify the information required for the sample application to connect to BRM

The following tables provide:

- A list of the sample programs
- A description of each sample program
- Information on any executable program that you can run to observe the results

Table 8–11 lists the file for setting makefile macros.

Table 8–11 Setting Makefile Macros (File Located in BRM_SDK_Home/source/samples)

Sample	Description
env.unix	Shows you how the environment is set up, for example, the location of include directories. The makefiles reference the appropriate environment file for this information. Instructions on setting the makefile macros are included in these text files.

Table 8–12 lists the sample files for creating an flist.

Table 8–12 Creating an Flist (Files Located in BRM_SDK_Home/source/samples/flists/C++)

Sample	Description
simple_flist.cpp	Shows how to create an flist with simple fields. Run **simple_flist.exe** to see a printout of the flist created, which contains a POID and two strings containing the first and last names. For information on how to run **simple_flist**, see "Running the Sample PCM C Programs".
flists_with_arrays.cpp	Shows how to create flists with arrays containing a single element and multiple elements. Run **flists_with_arrays.exe** to see the flists created by this sample. For information on how to run **flists_with_arrays**, see "Running the Sample PCM C Programs".
flists_with_substruct.cpp	Shows how to create an flist with a substructure. Run **flists_with_substruct.exe** to see the flists created by this sample. For information on how to run **flists_with_substruct**, see "Running the Sample PCM C Programs".

Table 8–13 lists the sample file for creating a context.

Table 8–13 Creating a Context (File Located in BRM_SDK_Home/source/samples/context/C++)

Sample	Description
create_context.cpp	Shows you how to open a context, connect to BRM, perform operations, test if the connection is open, and close the context. Run **create_context.exe** to verify that the program returns input flist as the output. For information on how to run **create_context**, see "Running the Sample PCM C Programs".

Table 8–14 lists the sample file for calling an opcode.

Table 8–14 Calling an opcode (File Located in BRM_SDK_Home/source/samples/callopcode/C++)

Sample	Description
test_loopback.cpp	Shows you how to call an opcode. This sample calls the PCM_OP_TEST_LOOPBACK opcode which just returns the flist that you pass in as the input. Run **test_loopback.exe** to verify that the program returns input flist as the output. For information on how to run **test_loopback**, see "Running the Sample PCM C Programs".

Table 8–15 lists the sample files for creating a client application.

Table 8–15 Creating a Client Application (Files Located in BRM_SDK_Home/source/samples/apps/C++)

Sample	Description
sample_act.cpp	Shows how to generate activity for a service. This sample generates email activity for an account. Run **sample_act.exe** to see how the program works. For information on how to run **sample_act**, see "Running the Sample PCM C Programs".
sample_PinBD.cpp	Shows how to use the class PinBigDecimal. This program illustrates how to create a big decimal number from a string or double, the use of various rounding modes and setting the number of decimal places, the use of mathematical functions, etc. Run **sample_PinBD.exe** to see how the program works. For information on how to run **sample_PinBD**, see "Running the Sample PCM C Programs".

Table 8–16 lists the sample files for using the multithreaded application (MTA) APIs.

Table 8–16 Files for Using the Multithreaded Application (MTA) API

Sample	Description
pin_mta_monitor (located in *BRM_SDK_Home/bin*)	Sample monitoring utility.
pin_mta_test.c (located in *BRM_SDK_Home/source/samples/apps/c/mta_sample*)	Sample test program using the MTA framework.

Compiling the Sample PCM C++ Programs

In addition to using the sample programs as working programming examples, you can also use them as a basis for your own applications. You can make changes to the sample programs, compile, and run them to test your changes. The sample programs directory includes the following files:

- **env.unix** to set the environment
- Makefiles for UNIX to compile the samples

To compile the sample programs:

1. Go to *BRM_SDK_Home*/**source/samples**, and open **env.unix**.
2. Set up the path for the environment by following the instructions in the file.
3. Save the file.
4. Compile using the **make** utility:

```
make
```

Running the Sample PCM C++ Programs

The executable versions of the sample programs are provided. To see the output generated by a sample program, follow these basic steps:

1. Go to the directory where the sample program is located. The default path is *BRM_SDK_Home*/**source/samples**.

2. Edit the entry in the configuration file **pin.conf** to point to the CM.
3. Run the program by running the executable, for example:

```
create_context.exe
```

About Using the PCM Java Sample Programs

BRM SDK includes a set of sample applications using the PCM Java API. You can use these sample programs in the following ways:

- Use the sample programs as code samples for extending BRM components and applications and for writing custom applications.
- Run the corresponding executable application with a sample program to observe the changes it makes in BRM.

These samples are supported on several platforms: Linux, AIX, Solaris, and HP-UX IA64. Compile these sample programs using the appropriate compiler for your platform.

Finding the Sample PCM Java Programs

When you install BRM SDK, the sample programs are installed by default in *BRM_Home*/**InfranetSDK/source/samples.**

For information on installing BRM SDK, see "Installing BRM SDK" in *BRM Installation Guide*.

You can also display the sample programs by clicking the links in this document.

Note: The installation directory is called *BRM_SDK_home* in the documentation.

Description of the Sample PCM Java Programs

The sample programs demonstrate how to write code for various tasks when customizing BRM.

Each sample includes these supporting files:

- Source files to view or modify for your own applications
- Makefiles to compile the sample programs, if you make changes to the samples
- A compiled application that verifies that the sample programs work as expected and that allows you to observe the changes the programs make in BRM
- A configuration file **infranet.properties** that allows you to specify the information required for the sample application to connect to BRM

The following tables provide:

- A list of the sample programs and makefiles
- A description of each sample program and makefile
- Information on any executable program that you can run to observe the results

Table 8–17 lists the sample file for setting the makefile macros.

Table 8–17 Setting Makefile Macros (File Located in BRM_SDK_Home/source/samples)

Sample	Description
env.unix	Shows you how the environment is set up, for example, the location of include directories. The makefiles reference the appropriate environment file for this information. Instructions on setting the makefile macros are included in these text files.

Table 8–18 lists the sample files for creating an flist.

Table 8–18 Creating an Flist (Files Located in BRM_SDK_Home/source/samples/flists/Java)

Sample	Description
SimpleFlist.java	Shows how to create an flist with simple fields. Run **SimpleFlist.class** to see a printout of the flist created, which contains a POID and two strings containing the first and last names. For information on how to run **SimpleFlist**, see "Running the Sample PCM C Programs".
FlistsWithArrays.java	Shows how to create flists with arrays containing a single element and with arrays containing multiple elements. Run **FlistsWithArrays.class** to see the flists created by this sample. For information on how to run **FlistsWithArrays**, see "Running the Sample PCM C Programs".
FlistsWithSubstructs.java	Shows how to create an flist with a substructure. Run **FlistsWithSubstructs.class** to see the flists created by this sample. For information on how to run **FlistsWithSubstructs**, see "Running the Sample PCM C Programs".

Table 8–19 lists the sample file for creating a context.

Table 8–19 Creating a Context (File Located in BRM_SDK_Home/source/samples/context/Java)

Sample	Description
CreateContext.java	Shows you how to open a context, connect to BRM, perform operations, test if the connection is open, and close the context. Run **CreateContext.class** to see how to open a context. For information on how to run **CreateContext**, see "Running the Sample PCM C Programs".

Table 8–20 lists the sample file for calling an opcode.

Table 8–20 Calling an Opcode (File Located in BRM_SDK_Home/source/samples/callopcode/Java)

Sample	Description
TestLoopback.java	Shows you how to call an opcode. This sample calls the PCM_OP_TEST_LOOPBACK opcode which just returns the flist that you pass in as the input. Run **TestLoopback.class** to verify that the program returns input flist as the output. For information on how to run **TestLoopback**, see "Running the Sample PCM C Programs".

Table 8–21 lists the sample files for creating a client application.

Table 8–21 Creating a Client Application (Files Located in BRM_SDK_Home/source/samples/apps/Java)

Sample	Description
CreateCustomUsageEvent.java	Shows you how to generate an email activity event for a particular account. Run **CreateCustomUsageEvent.class** to see how the program works. For more information on **CreateCustomUsageEvent**, see "Creating Events by Using the CreateCustomUsageEvent.java Program". For information on how to run **CreateCustomUsageEvent**, see "Creating Events by Using the CreateCustomUsageEvent.java Program".
CreateCustomer.java	Shows you how to create a new customer through the user interface defined in **CreateCustomerUI.java,**using the account information definition from **CreateCustomerAccountInfo.java** and the model created by **CreateCustomerModel.java**. Run **CreateCustomer.class** to see how to create a customer using these four programs. For more information on **CreateCustomer**, see "Creating Accounts by Using the CreateCustomer.java Program". For information on how to run **CreateCustomer**, see "Running the Sample PCM C Programs".
CreateCustomerUI.java	Defines the user interface used by **CreateCustomer**.
CreateCustomerAccountInfo.java	Defines the account information and holds the data.
CreateCustomerModel.java	Shows you how to create new customers by creating flists to pass information to it, including customer name and address, pertinent plan, billing information, invoice data, etc. Then it adds the requested login and password to each service array element and creates the customer in the BRM database. Of the four **CreateCustomer** programs, **Create CustomerModel.java** is where all the BRM actions take place in this program..

Compiling the Sample PCM Java Programs

In addition to using the sample programs as working programming examples, you can also use them as a basis for your own applications. You can make changes to the sample programs, compile, and run them to test your changes. The sample programs directory includes the following files:

- **env.unix** to set the environment
- Makefiles to compile the samples

To compile the sample programs:

> **Important:** To compile the sample programs, you must have a Java compiler installed on your system. For a list of compatible versions of the Java compiler, see "BRM Software Compatibility" in *BRM Installation Guide*.

1. Go to *BRM_SDK_Home*/**source/samples**, and open **env.unix**.
2. Set up the path for the environment by following the instructions in the file. Make sure the JDK_HOME variable includes the absolute path of your Java compiler.
3. Save the file.
4. Compile using the **make** utility:

```
make
```

Running the Sample PCM Java Programs

The executable versions of the sample programs are provided. To see the output generated by a sample program, follow these basic steps:

1. Go to the directory where the sample program is located. The default structure is: *BRM_SDK_Home*/**source/samples**.
2. Edit the configuration file **infranet.properties** to point to the CM.
3. Set the classpath to:

   ```
   java -classpath <path to jar files> <sample_name>
   ```

   For example:

   ```
   classpath/BRM_SDK_Home/jars/pcm.jar;/BRM_SDK_Home/jars/pcmext.jar;. SimpleFlist
   ```

4. Run the program, for example:

   ```
   java create_context
   ```

Creating Accounts by Using the CreateCustomer.java Program

The **CreateCustomer.java** program creates a new account with services in the specified plan. You can modify this program to add new services to an account or to create dummy accounts to test BRM functionality.

This program performs the following actions:

1. Opens a database channel
2. Retrieves the specified plan
3. Adds the customer information to the plan
4. Creates the customer account
5. Closes the database channel

For information on the structure and parameters, look at the source file **CreateCustomer.java** located in *BRM_SDK_Home*/**source/samples/apps/Java**.

Creating Events by Using the CreateCustomUsageEvent.java Program

The **CreateCustomUsageEvent.java** program simulates customer activity by creating an activity event for an email service object. Use this program to generate any number of email events.

For information on the structure, see the source file **CreateCustomUsageEvent.java** located in *BRM_SDK_Home*/**source/samples/apps/Java**.

Running the CreateCustomUsageEvent Program

1. Create the storable class of type **event/activity/email** and these custom fields.

   ```
   EMAIL_EVENT_INFO   PIN_FLDT_SUBSTRUCT [0]   ID# 10001
   EMAIL_FROM    PIN_FLDT_STR [0]             10002
   EMAIL_TO      PIN_FLDT_STR [0]             10003
   ```

   For information, see "Creating, Editing, and Deleting Fields and Storable Classes" in *BRM Developer's Guide*.

2. Follow the instructions in "Making Custom Fields Available to Your Applications" in *BRM Developer's Guide* to make the custom fields available to your applications.
3. Restart the CM, the client tools, and other components.
4. Run **CreateCustomUsageEvent** to generate email activity events:

```
java CreateCustomUsageEvent
```

About Using the PCM Perl Sample Programs

BRM SDK includes a set of sample applications using the PCM Perl API. You can use these sample programs in the following ways:

- Use the sample programs as code samples for extending BRM components and applications and for writing custom applications.
- Run the corresponding executable application with a sample program to observe the changes it makes in BRM.

These samples are supported on several platforms: Linux, AIX, Solaris, and HP-UX IA64. Compile these sample programs using the appropriate compiler for your platform.

Finding the Sample PCM Perl Programs

When you install BRM SDK on UNIX, the sample programs are installed by default in *BRM_Home*/**InfranetSDK/source/samples**.

For information on installing BRM SDK, see "Installing BRM SDK" in *BRM Installation Guide*.

You can also display the sample programs by clicking the links in this document.

Note: The installation directory is called *BRM_SDK_home* in the documentation.

Description of the Sample PCM Perl Programs

The sample programs demonstrate how to write code for various tasks when customizing BRM.

Each sample includes these supporting files:

- Source files to view or modify for your own applications
- A compiled application that you can run to verify that the sample programs work as expected and to observe the changes the program makes in BRM
- A configuration file **pin.conf** where you specify the configuration information for the sample application to connect to BRM

The following tables provide:

- A list of the sample programs
- A description of each sample program
- Information on any executable program that you can run to observe the results

Table 8–22 lists the sample files for creating an flist.

Table 8–22 Creating an Flist (Files Located in BRM_SDK_Home/source/samples/flists/perl)

Sample	Description
simple_flist.pl	Shows how to create an flist with simple fields. Run simple_flist.pl to see a printout of the flist created, which contains a POID and two strings containing the first and last names.
flist_with_arrays.pl	Shows how to create flists with arrays containing a single element. Run **flist_with_arrays.pl** to see the flist created by this sample.
flist_with_substruct.pl	Shows how to create an flist with a substructure. Run **flist_with_substruct.pl** to see the flist created by this sample.

Table 8–23 lists the sample files for creating a context.

Table 8–23 Creating a Context (Files Located in BRM_SDK_Home/source/samples/context/perl)

Sample	Description
connect.pl	Shows you how to open a context, connect to BRM using **pin.conf** parameters, perform operations, test if the connection is open, and close the context. Run **contect.pl** to verify that the program returns input flist as the output. For information on how to run **connect.pl**, see "Running the Sample PCM C Programs".
create_context.pl	Shows you how to open a context, connect to BRM using logon information within the program, perform operations, test if the connection is open, and close the context. Run **create_context.pl** to demonstrate how to open a context. For information on how to run **create_context.pl**, see "Running the Sample PCM C Programs".

Table 8–24 lists the sample file for calling an opcode.

Table 8–24 Calling an Opcode (File Located in BRM_SDK_Home/source/samples/callopcode/perl)

Sample	Description
test_loopback.pl	Shows you how to call an opcode. This sample calls the PCM_OP_TEST_LOOPBACK opcode which just returns the flist that you pass in as the input. Run **test_loopback.pl** to verify that the program returns input flist as the output. For information on how to run **test_loopback.pl**, see "Running the Sample PCM C Programs".

Running the Sample PCM Perl Programs

The executable versions of the sample programs are provided. To see the output generated by a sample program, follow these basic steps:

1. Go to the directory where the sample program is located. The default structure is: *BRM_Home*/**InfranetSDK/source/samples**.
2. Edit the entry in the configuration file **pin.conf** to point to the CM.
3. Run the program by executing the program name under Perl, for example:

```
perl create_context.pl
```

Note: Use the Perl installed by the SDK (or with the BRM server), located in *BRM_Home***/perl/bin/perl**. This version of Perl is preconfigured for BRM.