Oracle® Communications Billing and Revenue Management

Managing Accounts Receivable

Release 7.5

E16695-10

April 2017

ORACLE®

Oracle Communications Billing and Revenue Management Managing Accounts Receivable, Release 7.5

E16695-10

Contents

Preface

This book describes how to use and manage accounts receivable (A/R) data in Oracle Communications Billing and Revenue Management (BRM).

Audience

This book is intended for operations personnel and system administrators.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Technology Network:

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
E16695-01	November 2011	Initial release.
E16695-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ■ Made minor formatting and text changes.

Version	Date	Description
E16695-03	August 2012	Documentation updates for BRM 7.5 Patch Set 2. ▪ Made minor formatting and text changes.
E16695-04	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ▪ Updated "About Balance Groups" for policy-driven charging.
E16695-05	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ▪ Replaced multidatabase information with multischema information.
E16695-06	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ▪ Updated "Specifying Search Criteria for Bill Items". ▪ Updated "About the Item Data Retrieved". ▪ Updated "About the Item Detail Data Retrieved".
E16695-07	August 2014	Documentation updates for BRM 7.5 Patch Set 9. ▪ Updated "About Hierarchical Bill Units". ▪ Updated "Creating Hierarchical Bill Units". ▪ In the "pin_mass_refund" utility section, updated the description of **pay_type** parameter to include SEPA. ▪ In the "pin_refund" utility section, updated the syntax, changed vendor to optional. Also updated the description of **pay_type** parameter to include SEPA.
E16695-08	October 2014	Documentation updates for BRM 7.5 Patch Set 10. ▪ Updated "Enabling Automatic Write-Off Reversals during Payment Collection". ▪ Added "Specifying the Level of Write-Offs for Reversals". ▪ Added information about write-off reversals at the bill-unit level in multiple sections.
E16695-09	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ▪ Added the "Configuring BRM to Get Events based on Balance Impact Type" section. ▪ Updated the "Configuring Event Notification for Balance Monitoring" section.
E16695-10	April 2017	Documentation updates for BRM 7.5 Patch Set 18. ▪ Added the following sections: Adjusting Fully or Partially Paid Bills Improving Bill-Level Adjustments for Fully or Partially Paid Bills ▪ Updated the following sections: Adjusting Bills Fields You Should Include in the Flist for Bill Adjustments Tax Processing for Bill and Item Level Adjustments

1

About Accounts Receivable

This chapter describes how Oracle Communications Billing and Revenue Management (BRM) stores, displays, and manipulates accounts receivable (A/R) data.

Caution: Do not use or modify this product except as explicitly instructed in this documentation. Assumptions should not be made about functionality that is not documented or use of functionality in a manner that is not documented. Use or modification of this software product in any manner or for any purpose other than as expressly set forth in this documentation may result in voidance or forfeiture of your warranties and support services rights. See your software license agreement for more details. If you have any questions regarding an intended use or modification of this product, contact Oracle.

For information about the procedures for managing payments, see "About BRM-Initiated Payment Processing" and "Managing Externally Initiated Payments" in *BRM Configuring and Collecting Payments*.

For information about configuring adjustments, disputes, and settlements, see "Configuring Adjustments, Disputes, and Settlements".

For basic information about BRM and billing, see the discussion on BRM in *BRM Concepts* and "About Billing" in *BRM Configuring and Running Billing*.

How BRM Stores Accounts Receivable Information

When a customer generates a billable event, the BRM system rates the event to determine how much to charge the customer for it.

The BRM system stores the cost of the event, along with the cost of similar events generated by the customer, in an item in the BRM database. The cost of the events stored in the items affects the current balances of the customer’s account.

During billing, the BRM system collects the items for an account in a bill, which is also stored in the database.

The BRM billing utilities request payment for the bill from the customer.

Until payment is received, the money a customer owes, along with the amounts due from other customers, represents your company’s accounts receivable.

A/R actions impact the amount a customer owes your company. The effects of these A/R actions are stored in items. For more information, see "About Items".

About A/R Database Objects

To bill an account, BRM must track and store the following information for each billing cycle:

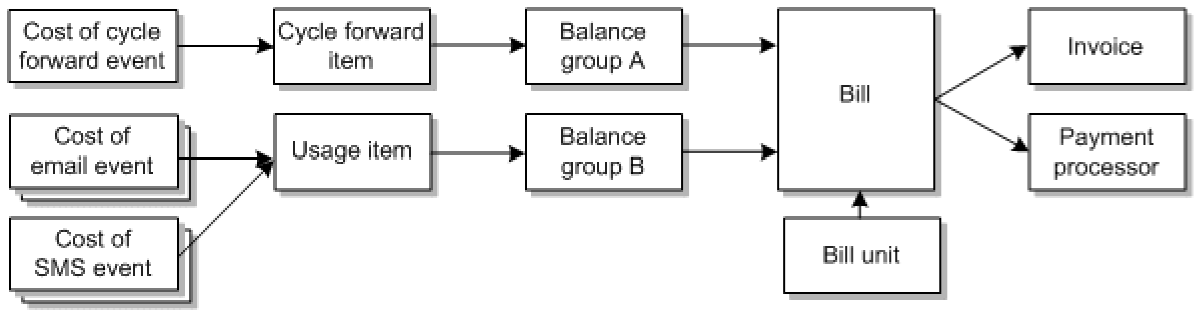
- **The balance impact for a billable event.** This information is stored in an **/item** object. Billable events are grouped by type. See "About Items".
- **Total balance due from all billable events in a billing cycle.** The total balance is stored in a **/balance_group** object. You can create multiple balance groups to track balance totals by service type. For example, one balance group could store the total balance of all GSM charges and another balance group could store the total balance of all GPRS charges. See "About Balance Groups".
- **When and how to generate a bill.** All administrative information for creating a bill, such as the payment method, billing cycle length, payment collection date, and account hierarchy information, is stored in the **/billinfo** object. See "About Bill Units".
- **The bill itself.** All information necessary to create a payment request is stored in a **/bill** object. See "About Bills".

When you are ready to request a payment from the customer, the billing utilities read the **/bill** object and generate one of the following:

- An invoice that can be mailed or emailed to the customer.
- A credit card transaction that is sent through a credit card processor.
- A direct debit transaction that is sent through a payment processing service.

Figure 1–1 shows a typical event to invoice billing cycle.

Figure 1–1 Billing Cycle



For example, a customer accumulates the following charges in one billing cycle: a $20 cycle forward fee, a $10 email usage fee, and a $40 SMS text messaging usage fee. BRM stores the A/R data in the following objects:

- One **/item** object stores the $20 cycle forward fee and another **/item** object stores the $50 in usage fees.
- A **/balance_group** object stores the total cost of all bill items ($70).
- A **/billinfo** object stores information about when and how to generate a bill. For example, it might specify the 5th as the billing day of month (DOM), the next billing date as January 5, and an invoice payment method.
- A **/bill** object is generated on January 5 that includes all billing information, such as a list of billable events that occurred during the cycle, any adjustments or

credits made by customer service representatives (CSRs), any customer payments, and the total amount due.

When you are ready to request a payment, you run the appropriate BRM utility to generate a printed invoice and then mail the invoice to the customer.

About Items

Item objects in the BRM database store A/R data for an account.

Types of Items

To understand BRM accounts receivable, think of two types of items: bill items, which contribute to the total amount stored in a bill, and A/R items, which reflect actions on the A/R of an account or account bill, such as payments or adjustments. BRM manages A/R by transferring amounts between A/R items and bill items. See "How A/R Actions Affect Account Balances".

Bill Items

Bill items include cycle forward items, cycle arrears items, usage items, and custom items. See "About Bill Items" in *BRM Configuring and Running Billing*.

A/R Items

A/R items include adjustment items, dispute items, settlement items, payment items, refund items, payment reversal items, write-off items, and write-off reversal items. A/R items collect balance impacts from the type of event reflected in their name.

For information on the general ledger (G/L) impact of the events that contribute to bill items and A/R items, see See "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data*.

Fields in an Item

All item objects in the BRM database include the same fields.

For a technical description of item objects, see **/item** in *BRM Storable Class Reference*.

The following item fields are of particular importance in understanding the flow of amounts through the BRM accounts receivable system:

You can think of the Due, Adjusted, Disputed, Transferred, and Received fields as buckets that contain coins. As the BRM system manipulates A/R, it moves coins among these buckets, but it never changes the number of coins in the Total bucket. The Total bucket contains the coins from all the events linked to the item.

Figure 1–2 shows an example account's total charges in a cycle. The customer has accumulated a hundred dollars ($100) in charges. Without any adjustments the customer has a balance due of $100.

Figure 1–2 Total Cycle Charges

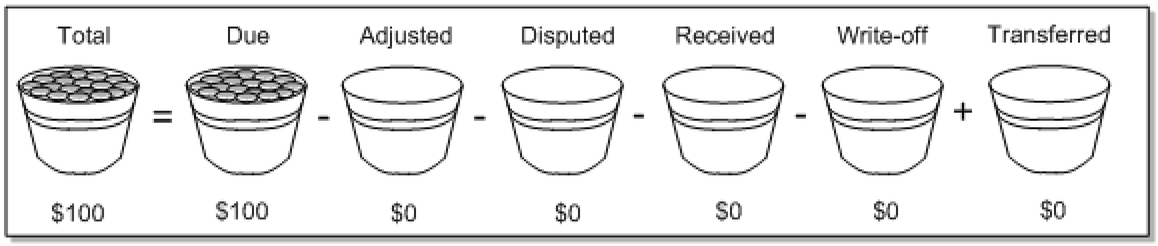
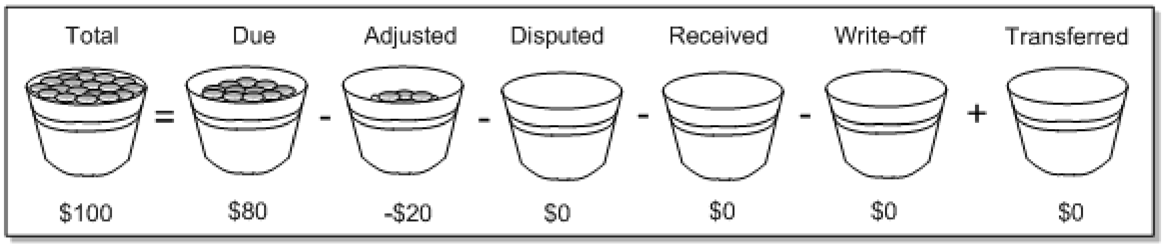


Figure 1–3 shows the same account holder's total charges of $100. An adjustment of $20 has reduced the customer's balance due to $80. The total value of the balance due and the adjustment still equals the total amount of charges.

Figure 1–3 Balance Due Impacted by Adjustment



- **Total**: The sum of all balance impacts to a currency resource that occurred as a result of the events that contribute to the item. For example, the usage item stores the costs of the usage events a customer generated during the accounting cycle.
- **Due**: The amount owed by the customer for this item. Initially, Due is the same as Total, but after A/R is received from or transferred to another item, the amounts may be different. For example, the Total and Due of a bill item are the same when the item is created, but payments, adjustments, or disputes reduce the Due.

Note: The amount due includes the balance impact of unresolved disputes.

- **Adjusted**: The total amount of adjustments made to the item, the net of both debit and credit adjustments. Because most adjustments are credits, the amount in the adjustment field usually reduces the Due of the item.
- **Disputed**: The total amount of unresolved disputes against the item. The disputed amount reduces the Due of the item.
- **Received**: The total amount transferred into this item from another item, which is usually a payment item.
- **Write-off**: A write-off is an A/R transaction that removes an uncollectable balance from a customer's account so it is not considered as an asset for accounting purposes.
- **Transferred**: The total amount transferred out of this item and into another item.
- **Status**: The state of the item, which can be Pending (unbilled), Open (billed), or Closed (zero amount due). Items generally move from Pending to Open to Closed status, but closed items can be reopened.

When thinking about item fields, remember that:

- Adjustment and dispute *items* are different from the Adjusted and Disputed *fields* in an item. Adjustment and dispute items have Adjusted and Disputed fields, although these fields are unlikely to store a value other than zero for these items.
- A transfer operation places values in the Transferred field of the source item and the Adjusted, Disputed, Received, or Write-off field of the target item, based on the A/R operation being performed.
- Credit values are stored as negative numbers.
- The Status field represents the condition of the item in the flow of A/R.

About Balance Groups

A balance group is an object in the BRM database that stores the balance total for a group of bill items. For example, **/balance_group** objects store the following information:

- Account associated with the balance group.
- Bill unit associated with the balance group.
- Current balance total.
- Balance that is currently reserved for an active prepaid session. See "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.
- Consumed reservation amount.
- Details about any sub-balances, such as the sub-balance total and validity dates. See "About Tracking Resources in Account Sub-Balances" in *BRM Setting Up Pricing and Rating*.
- Resource consumption rule. See "Specifying the Order in Which Resource Sub-Balances Are Consumed" in *BRM Setting Up Pricing and Rating*.

All customer accounts in the database, including accounts that do not pay their own bills, have their own account-level **/balance_group** object. Customers can optionally add service-level balance groups to track the balances for specific services. See "Tracking Resources by Service" in *BRM Setting Up Pricing and Rating*.

Bill items for an account are stored for each account balance group. During billing, items for each balance group are collected in the bill.

Balance groups contain fields that store sub-balance data, such as the type of resource stored in the sub-balance, the amount in each sub-balance, and the dates when the resource is valid. The sub-balances are grouped by the type of resource they track.

For a technical definition of the balance group object, see **/balance_group** in *BRM Storable Class Reference*.

About the Default Balance Group of a Bill Unit

In general, BRM applies event balance impacts to the balance group of the service for which the event was generated. However, some events are not generated for any service. For example, bill-level adjustment and dispute, payment, payment incentive, and payment fee events are generated for a specific bill unit. In such instances, BRM applies the event balance impact to the default balance group of the bill unit and assigns the event to an item belonging to the default service of the default balance group. See "About the Default Service of the Default Balance Group".

Customer Center chooses a random balance group to be the default for new bill units during account creation and account maintenance. You cannot choose a different default balance group for a bill unit.

Note: During account maintenance through Customer Center, bill units that already have a default balance group are not changed.

If you use the PCM_OP_CUST_CREATE_CUSTOMER and PCM_OP_CUST_MODIFY_CUSTOMER opcodes to create and modify accounts, you must pass the default balance group array for each bill unit to the opcodes.

About the Default Service of the Default Balance Group

BRM uses the default service of the default balance group primarily to assign events that are not associated with any service to the items of the default service. See "About the Default Balance Group of a Bill Unit".

BRM selects a default service using the same algorithm for selecting the default balance group. It selects the service object in the default balance group with the smallest create time (PIN_FLD_CREATED_T). If more than one service object is selected, the service object with the smallest POID is selected as the default service of the default balance group.

You can override the algorithm BRM uses to select the default service. You do this by adding your custom algorithm to the PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC policy opcode.

About Bill Units

A *bill unit* is a **/billinfo** object in the BRM database that stores information on how, how often, and when to generate a **/bill** object. For example, a bill unit stores the following information:

- Billing state: open, closed, or done.
- Billing status: active, inactive, or accounting only.
- Billing DOM.
- Billing type: open-item accounting or balance-forward accounting.
- Starting date of the current billing cycle.
- Starting date of the next billing cycle.
- Billing frequency, in number of months.
- Primary currency.
- Hierarchical information, such as if the bill unit is a subordinate or a parent account.
- Payment method: credit card, invoice, direct debit, or paid by parent account.
- Payment collection date.
- Default balance group associated with the bill unit.

Note: A bill unit must be associated with at least one balance group.

- Account associated with the bill unit.
- Bill associated with the bill unit.

BRM creates one **/bill** object for each **/billinfo** object in the BRM database. All customer accounts in the database, including accounts that do not pay their own bills, have their own **/billinfo** objects. Customers can optionally add additional **/billinfo** objects to their account to divide charges into separate bills. For example, a customer can have one **/billinfo** object that generates a bill for all services related to a home-based business and another **/billinfo** object that generates a bill for all personal telephony charges.

For a technical definition of the bill unit object, see **/billinfo** in *BRM Storable Class Reference*.

About Bills

A BRM bill is a **/bill** object in the BRM database that stores all billing information necessary to generate a request for payment. BRM creates one **/bill** object for each **/billinfo** object in the BRM database.

The **/bill** object only *stores* billing information and is not a request for payment itself. You request payments from customers by creating the following:

- Invoices, which are generated by the **pin_inv_accts** utility (see "pin_inv_accts" in *BRM Designing and Generating Invoices*)
- Credit card or debit card transactions, which are generated by the **pin_collect** utility (see "pin_collect" in *BRM Configuring and Collecting Payments*)

All customer accounts in the database, including accounts that do not pay their own bills, have their own **/bill** objects. The **/bill** object stores data such as the bill creation date, total charges accumulated in the bill, the amount due from the customer, and the bill due date.

For a technical definition of the bill object, see **/bill** in *BRM Storable Class Reference*.

About Generating Multiple Bills per Cycle for a Single Account

You can generate multiple bills per billing cycle for a single account. You might do this to:

- Permit a customer to pay for different services by using different payment methods, such as credit card for one service and invoice for another.
- Track usage and bill customers for individual subscriptions rather than their account. For more information, see "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.

When billing is run, a bill is generated for each **bill unit** (**/billinfo** object) owned by the account.

By default, accounts are created with one account-level **/billinfo** object. You can create, adjust, and modify the default account-level **/billinfo** object by using Customer Center.

Customizing How to Modify a Bill Object

To modify a bill object before it is committed in the database, use the PCM_OP_BILL_POL_BILL_PRE_COMMIT policy opcode.

This policy opcode is called by the PCM_OP_BILL_MAKE_BILL opcode when a **/bill** object is created.

About Bill States

The **/billinfo** object stores the state of a bill and can have one of the following values:

- **0** to indicate that the bill has been finalized.
- **1** to indicate that partial billing has been completed.
- **2** to indicate that all cycle charges have been applied at the end of the billing period and the bill needs to be finalized.

When an account with best pricing configuration is billed, the bill advances to the next state, except in certain cases such as triggered billing and billing for previously skipped months.

About A/R Management

A/R actions constitute the major A/R management activities. A/R actions include recording externally initiated payments and payment reversals, making adjustments, creating and settling disputes, issuing refunds, and writing off bad debt.

CSRs perform many A/R actions in the **Balance** tab of Customer Center. A CSR must have the proper permission to perform A/R actions and adjust account balances in Customer Center. See "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide*.

To properly account for the effects of A/R actions on your company's revenue, be sure to assign G/L IDs to the associated events. See "Assigning G/L IDs to Nonrated Events" in *BRM Collecting General Ledger Data*.

About Viewing A/R Information

A CSR can see the balances of an account's currency and non-currency resources in Customer Center. The Customer Center **Balance** tab displays this information. See "Balance Tab" in Customer Center in *BRM Managing Customers*.

The Customer Center **Balance** tab displays the billed, unallocated, and unbilled amounts included in the current balances of each account and the nonpaying bill unit of its child accounts. If the current account is a child account with a nonpaying bill unit, you see information for its A/R parent account.

For accounts that use two currencies, you can display account balances in either the primary account currency or secondary account currency in Customer Center.

If your company has implemented a customer self-care Web site, customers can see the balance of their accounts on the Web. See "Ways to Implement a Web Interface" in *BRM Managing Customers*.

For information on viewing A/R information in Customer Center, see the Customer Center Help.

Determining the Amount Due

The **Summary** tab of Customer Center displays the sum of the due amounts for all bills of the account; the *total due amount*. The total due amount is reduced by unallocated payments and adjustments. To see the due amount for a bill, a service included in a bill, or a bill item, click the **Balance** tab.

Exporting A/R Data to Files

You can export the data in these Customer Center tables to an HTML file:

- Balance information.
- Payments, A/R actions, and item charges.

Displaying the History of an Item

An item's history shows details about the amounts the item received from and transferred to other items. This information is available for both automatic transfers performed by BRM and manual transfers performed by using Customer Center. See "Transferring Amounts between Items".

Note: For information on how to display the history of an item in Customer Center, see Reviewing bill item charges and details.

Configuring BRM to Get Events based on Balance Impact Type

You can limit the event search criteria to just one type of balance impact by changing the value of the new PIN_FLD_MODE field directly or by setting the EventChargeDiscountMode parameter to one of the following values:

- **0:** (Default and only pre-fix mode) Returns events whose gross, net, or total balance impact matches the search criteria.
- **1:** Returns events whose gross balance impact (the sum of all charges) matches the search criteria. Discounts and tax are not considered.
- **2:** Returns events whose net balance impacts (the sum of all charges and discounts) matches the search criteria. Tax is not considered.
- **3:** Returns events whose total balance impact (the sum of all charges, discounts, and tax) matches the search criteria.

Improving Item Search Performance

If an A/R account has subordinate accounts, BRM performs a step search to find items. By default, BRM returns 1000 items with each step of the search.

To customize the number of items returned:

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

   For example, to set the number of returned items to 2000:

   ```
   - fm_bill     item_search_batch     2000
   ```

   You can also choose not to use step-search by setting the number of returned items to 0.

   Important: If your A/R account has a large hierarchy, bypassing step-search can cause the Data Manager (DM) to run out of memory.

Use larger batch search memory sizes to improve run-time performance. You do not need to restart the CM to enable this entry.

How A/R Actions Affect Account Balances

When the BRM system creates a bill item, the Total and Due fields of the item are equal. When an A/R action occurs, the balance impact of the action is stored in an A/R item. For most A/R actions, the balance impact is immediately transferred to a bill item, which changes the Due of the bill item. For example, a payment typically reduces the Due of a bill item.

You can review the details of the transfers between bill items and A/R items by using Customer Center. See "Displaying the History of an Item".

When the Due of a bill item is reduced, the current balance of the account's balance group is also reduced. When the Due of the bill item reaches zero, the item is closed and no further requests for payment are made for it unless further A/R actions occur on the item and reopen it.

However, if the item is under dispute (that is, the disputed field is nonzero) the bill item would remain open even if the Due becomes zero.

Backdating A/R Actions

You can backdate A/R actions so that, for accounting purposes, the action is considered to have occurred at an earlier point in time and the revenue for that time is reported accurately. The A/R actions you can backdate include adjustments, disputes, settlements, externally initiated payments, payment reversals, and write-offs.

Note: BRM does not support future dating of A/R actions.

To backdate an A/R action, the CSR sets the transaction date (the date that the action is to take effect) to a date before the current date. When backdating, the CSR must select a date after the following:

- The last posted G/L transaction report. See "Posting G/L Reports" in *BRM Collecting General Ledger Data*.
- The account creation date.

Whether backdated A/R actions appear on the current bill depends on the accounting method you use:

- If you use balance forward accounting, the total of A/R actions for all prior open bills appears on the current bill or invoice as part of the previous balance.
- If you use open item accounting, A/R actions do not appear on the current bill or invoice.

If customers request an invoice that includes information on specific A/R actions in a prior billing period, you can create the invoice by using the **testnap** utility to run the PCM_OP_INV_MAKE_INVOICE opcode. For information on **testnap**, see "Testing Your Applications and Custom Modules" in *BRM Developer's Guide*.

About Payments

In the simple case, payments are the easiest A/R action to understand. When a payment event occurs, the BRM system creates a payment item. The values in the **Total** and **Due** fields of the item are initially equal. If the payment is recorded against a bill item, the credit in the Due field of the payment item is immediately transferred to the bill item. The Due of the payment item is then zero and the payment item is closed.

After the transfer from the payment item, the Due field of the bill item is reduced by the amount of the payment. If the payment amount is the same as the amount of the bill item, the bill item is closed.

When a customer's payment is allocated to a bill item, the payment amount also reduces the current balance of the customer's account balance group. Unallocated payments also reduce the current balance of the customer's bill, but they do not reduce the Due of any bill item. The Due amounts of bill items continue to contribute to the Due of the bill and the customer continues to receive requests to pay the unallocated amount.

A payment clerk or CSR allocates payments for invoice accounts by using Payment Tool or Customer Center. For more information, see "About Allocating Payments" in *BRM Configuring and Collecting Payments*.

Payments made by credit card or direct debit are automatically allocated during the collection process. For information, see "About BRM-Initiated Payment Processing" and "Managing Externally Initiated Payments" in *BRM Configuring and Collecting Payments*.

For more information about payments and payment processing, see "About Payments" in *BRM Configuring and Collecting Payments*.

About Reversing Payments

When a payment has been recorded in the BRM database but has not yet been deposited, you can reverse it from the BRM system. This enables BRM to treat the payment as if it never happened. When a payment is reversed, any bills or items previously closed by the payment are reopened, and the payment is deactivated in the BRM system.

For more information, see "About Reversing Payments" in *BRM Configuring and Collecting Payments*.

About Suspending Payments

If you have the optional Payment Suspense Manager feature enabled, and payments fail validation when they are submitted to BRM, they can be suspended and saved to the payment suspense account. This enables you to continue with your payment processing without interruption and correct the suspended payments at a later time. After you correct suspended payments (for example, by providing the correct account number or bill number), you can submit them to one or more customer accounts.

For more information, see "Configuring Payment Suspense Manager" in *BRM Configuring and Collecting Payments*.

About Adjustments

A CSR usually performs an adjustment to satisfy an unhappy customer or correct a problem. For example, a CSR might give an adjustment when your company charged the entire monthly fee for a service that was unavailable for a few days. A/R adjustments usually credit the balance of currency resources in the customer's account and reduce the amount the customer owes. They do not return money directly to the customer as refunds. See "About Refunds".

A CSR can adjust the customer's account, subscription services, member services, bills, bill items, and selected events.

Depending on the type of adjustment (account, bill, and so forth), a CSR can adjust currency resources, non-currency resources, or both, as shown in Table 1–1:

Table 1–1 Adjustable Resources

Adjustment Type	Currency	Non-currency
Account-level adjustment[1]	Yes	Yes
Subscription service-level adjustment [1]	Yes	Yes
Member service-level adjustment [1]	Yes	Yes

Table 1–1 (Cont.) Adjustable Resources

Adjustment Type	Currency	Non-currency
Bill-level adjustment	Yes	No
Item-level adjustment	Yes	No
Event-level adjustment	Yes	Yes

[1] For these adjustments, there must be a non-currency balance group at the account level for the adjustment to affect a non-currency resource.

Credit adjustments for currency resources decrease the Due of a bill item or, for non-currency resources, increase the balance of the adjusted resource. If a CSR is crediting an event, the balance impact of that event is removed from the customer's account. Debit adjustments have the opposite effect.

BRM processes adjustments and records the adjustment's balance impact in slightly different ways for each type of adjustment. For detailed information on these differences, see "About Adjustments".

Making Adjustments through Customer Center

CSRs make adjustments in Customer Center. A CSR can make either a credit or debit adjustment. For example, if a customer questions an hourly charge, the CSR can credit the amount at issue.

CSRs can adjust accounts, subscription services, member services, bills, bill items, and events. For all levels except the bill and bill item level, the CSR can adjust both currency and non-currency resources. Bill and bill item adjustments are for currency only.

CSRs can make adjustments at each of these levels in Customer Center; they can also adjust events in the Event Browser. To make adjustments in Customer Center, the CSR uses the **Balance** tab.

Note: For information on how to perform adjustments in Customer Center, see the Customer Center Help.

An account, subscription service, and member service adjustment to a currency resource sets aside an amount that must be allocated to a bill item. Use Customer Center to allocate the adjustment. Until you do this, account-level, subscription service-level, and member service-level credit adjustments reduce the current balance in the customer's account but they do not reduce the amount a customer is asked to pay.

For account, subscription service, or member service adjustments, perform a transfer to allocate an adjustment of balances paid by credit card or to allocate a debit adjustment. See "Transferring Amounts between Items".

About Opening and Resolving Disputes

A CSR creates a dispute when a customer disagrees with the amount he or she is asked to pay and the problem requires investigation before it can be resolved. A settlement is the resolution of a dispute; a settlement might favor the customer, favor the company, or divide the disputed amount between them. Disputes and settlements credit or debit the currency or non-currency resources of a customer's account but do not return money to the customer directly.

As with payments and adjustments, the BRM system creates a dispute item when the CSR enters a dispute. The disputed amount typically reduces the Due of a bill and the current balances of the customer's account. BRM does not request payment for the disputed amount.

To resolve a dispute, the CSR specifies how much of the disputed amount to grant to the customer. In settling a dispute, the CSR can grant the entire disputed amount to the customer, grant only part of the disputed amount, or deny the dispute and again request payment for the entire disputed amount. The BRM system creates a settlement item for the amount that is not granted and transfers amounts between the settlement item and the disputed bill item. After the dispute resolution, the Due and the current balance in the customer's account balance group reflect what the customer owes.

When the dispute item is created, its Total and Due are initially equal, but the BRM system immediately transfers the credit in the Due field of the dispute item to the disputed bill item, makes the Due of the dispute item zero, and closes the dispute item.

After a settlement, the amount of the dispute that is granted to the customer appears in the Adjusted field of the disputed bill item and in the histories of the disputed item and the settlement item.

A CSR can dispute and settle the customer's bill, bill items, and selected events. Depending on the type of dispute or settlement (bill, bill item, or event), a CSR can dispute or settle currency resources, non-currency resources, or both as shown in Table 1–2.

Table 1–2 Disputable Settlement Types

Dispute or Settlement Type	Currency	Non-currency
Bill-level dispute or settlement	Yes	No
Item-level dispute or settlement	Yes	No
Event-level dispute or settlement	Yes	Yes

Credit disputes for currency resources decrease the Due of a bill or bill item. If the customer is disputing the currency resources of an event for credit, the dispute removes the balance impact of the disputed event from the account. Debit disputes have the opposite effect.

Important: Event disputes for non-currency resources are recorded but do not directly affect the resource balance until the dispute is settled. At that point, the balance of the disputed resource is increased by the amount of the settlement.

BRM processes disputes and settlements in slightly different ways for each type of dispute or settlement. For detailed information on these differences, see "About Disputes and Settlements".

For information on how to open and resolve disputes in Customer Center, see the Customer Center Help.

About Refunds

You can give customers a refund whenever your company owes them money. Unlike an adjustment, which credits the customer's account balances, a refund returns money that your company owes a customer directly to the customer. Refunds for BRM-initiated payments return money to the direct debit or credit card account the customer uses to make payments. For customers who pay invoices, your company makes a refund by check or other externally initiated payment method.

Note: You cannot refund suspended payments. For information on how refunds relate to payment suspense processing, see "How Direct Reversals and Refunds Relate to Suspense" in *BRM Configuring and Collecting Payments*.

The following situations might require a refund:

- When a customer pays too much on a bill.
- When a customer has been given a credit adjustment.
- When a dispute has been settled and the customer is owed some money.

Instead of giving a refund, you can apply the money owed the customer to an open bill. You typically make refunds only if the customer must receive the money directly.

Refunding a customer is accomplished in two steps: creating a refund item, and making the refund payment. There are different ways to complete each step.

For information on how to refund accounts in Customer Center, see the Customer Center Help.

For information on how to define which items in your system are nonrefundable, see "Defining Nonrefundable Items" in *BRM Configuring and Running Billing*.

Creating Refund Items

The procedure for creating refund items varies depending on whether you want to refund a single account or perform a mass refund.

Creating refund items is not required; you can handle refunds by transferring A/R amounts and allocating payments between A/R items. (See "Making Adjustments through Customer Center".) Therefore, the **pin_mass_refund** utility is, by default, not included in any billing script.

A CSR can initiate a refund to a customer whose account or bill has a credit balance.

Creating Refund Items for All Accounts with a Credit Balance

The **pin_mass_refund** utility creates a refund item for accounts that have a credit balance. See "pin_mass_refund".

To give refunds to customers, include the **pin_mass_refund** utility in any of the billing scripts, such as the **pin_bill_day** script. If you do, run the **pin_mass_refund** utility before you run the **pin_refund** utility. See "Running Billing Utilities" in *BRM Configuring and Running Billing*.

Making Refund Payments

The procedure for making refund payments to customers varies depending on whether they pay your company by using a BRM-initiated or externally initiated payment method.

Making BRM-Initiated Refund Payments

Run the **pin_refund** utility to give refunds to all accounts that use the specified payment method and which have refund items when the **pin_refund** utility is run. You do not specify start and end date parameters when you run this utility. See "pin_refund".

If you miss a billing day, the **pin_refund** utility processes on all existing refund items.

When to Run the pin_refund Utility

The **pin_refund** utility is included by default in the **pin_bill_day**, **pin_bill_week**, and **pin_bill_month** scripts. For information about the billing scripts, see "About the Billing Utilities" in *BRM Configuring and Running Billing*.

Important: When you use multiple clearing house vendors, you run this utility for each clearing house. See "Using More Than One Payment Processor" in *BRM Configuring and Collecting Payments*.

For information about editing the billing scripts, see "Running Daily Billing" in *BRM Configuring and Running Billing*.

Setting the Minimum Amount to Refund

By default, **pin_refund** does not refund amounts less than two dollars. To change the minimum amount, see "Specifying the Minimum Amount to Refund" in *BRM Configuring and Running Billing*.

Making Externally Initiated Refund Payments

To make refund payments to customers whose payment method is not BRM-initiated, first create the payments outside the BRM system and then record them in the BRM database by using Payment Tool. For example, write checks to the customers who are due a refund and whose payment method is invoice, and then submit a refund batch. See "Processing a Batch of Payments by Using Payment Tool" in *BRM Configuring and Collecting Payments*.

Tip: You can create a custom application that finds refund items and sends the amount and account identification to a check-writing program.

About Refund Items

Refunds create refund items that appear in Customer Center. In other respects, they are different from other A/R actions.

The BRM A/R system creates a refund item to contain the credit in a customer’s account. For example, if a balance in a customer’s account is $110 and the customer owes your company $100, the amount in the refund item is $10.

When creating a refund item, the BRM A/R system evaluates the open bill items and A/R action items for the account, transfers credit among the items to close as many as possible, creates a refund item to contain the remaining credit, and closes the remaining open items that had a credit in Due.

There are two ways to create refund items:

- Use Customer Center to create refund items for individual accounts. You can create refunds for accounts or bills.
- Use the **pin_mass_refund** utility to create refund items for all accounts that have a credit balance; that is, all accounts that your company owes money to. See "pin_mass_refund".

About Refunding Child Accounts in Account Hierarchies

When you refund a non-paying child account, the refund is applied to the parent account. BRM associates the refund item with the parent's balance group that has the greatest amount due. If the parent account has no amount due, the refund item is associated with the default balance group of the parent account's bill unit.

For more information about account hierarchies, see "About Account Groups".

Making Refunds

To make refunds for BRM-initiated payments, you first create refund items and then run the **pin_refund** utility to deposit the refunds and close the refund items. By default, the **pin_bill_day** script runs the **pin_refund** utility daily. See "About the Billing Utilities" in *BRM Configuring and Running Billing*.

To make refunds for externally initiated payments, you begin by creating the refund items and then make the refund payments by check or other externally initiated payment method. You then use Payment Tool to record the refund in the BRM database and close the refund items. See "Processing a Batch of Payments by Using Payment Tool" in *BRM Configuring and Collecting Payments*.

After the refund, the credits in account balances are reduced by the refund amount.

Canceling Refunds

If you refund a customer account by mistake, you cannot cancel or reverse the refund. Instead, use the PCM_OP_AR_ACCOUNT_ADJUSTMENT opcode to perform a debit adjustment for the refund amount.

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

About Transferring Services between Balance Groups

CSRs can transfer a customer's service from one balance group to another balance group when the customer purchases a new deal or wants to track a service's balance separately.

Important: Only custom client applications can be used to transfer services between balance groups. To implement this functionality in your custom client application, see "Transferring Services between Balance Groups by Using Custom Client Applications".

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

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

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

For example, assume that an account owns two services, email and Internet access, that are each assigned to their own balance group. On June 1, balance group A has a balance of $30, and balance group B has a balance of $20. On June 15, the account transfers the email service to balance group A. Balance group A still has a balance of $30, and balance group B still has a balance of $20 as shown in Table 1–3.

Table 1–3 Multiple Balance Groups Example

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

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

Table 1–4 shows how transferring services between balance groups impacts other BRM features:

Table 1–4 Impact of Transferring Services between Balance Groups

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

About Transferring Amounts between Items

Transfers are made to correct payments that were allocated incorrectly, to allocate a debit adjustment for any account, and to allocate an account adjustment for an account that pays by credit card. In Customer Center, a CSR can transfer amounts between bill items. Transfers can also be made from payment items and account adjustment items to bill items.

A CSR can make a credit or debit transfer of a currency resource.

For more information, see "Applying Debits and Credits".

For information on how to transfer amounts in Customer Center, see the Customer Center Help.

About Writing Off Bad Debt

To perform a write-off when your company no longer expects to be paid, use Customer Center. A CSR can write off:

- Top-level A/R accounts.
- Bills.
- Bill items.

A write-off removes from your company's assets an A/R amount that your company has determined the customer will never pay. A write-off can also remove an A/R amount that your company has decided it will not refund to the customer; for example, if the amount is very small or if you sent the customer a check that was returned because the customer moved without leaving a new address.

To perform a write-off, use Customer Center. A CSR can write off:

- **Top-level A/R accounts:** Write-offs are not available for a top-level A/R account if its Due is zero.
- **Bills:** Write-offs are not available for a bill if it is pending or its Due is zero.
- **Bill items:** Item level write-offs are allowed for pending and open items.

To ensure proper write-off and write-off reversal functionality, the account should be inactivated before it is written off.

When a CSR performs a write-off, the BRM A/R system creates a write-off item for the amount to be written off. It transfers the Due from the write-off item to the bill items being written off and closes the bill items.

Writing off uncollectable debt or an unpayable credit lets you remove it from the books and more accurately control how it is treated in your accounting and G/L reporting system. Depending on how your company has set up its G/L system, the amount written off is transferred from A/R to a bad debt account or, for unpayable credit, to a miscellaneous revenue account.

A write-off always includes taxes, but you can specify whether the net and tax amounts are written off in one or two events. If you write off in one event, the net and tax amounts are stored separately within the event and can be mapped to different sets of G/L accounts. If you write off in two events, one event stores the net amount of the write-off while another event stores the tax amount. The net and tax amounts stored in these two events can also be mapped to separate sets of G/L accounts. However, the way you specify mapping information for G/L accounts is different for the one-event and the two-event write-off. For more information on how you map the net and tax amounts of a write-off to separate sets of G/L accounts, see "Assigning G/L IDs to Nonrated Events."

If a write-off on an account must be reversed, you can perform a write-off reversal by calling the write-off reversal opcodes. This returns the written-off amount to the account balance.

For more information, see "About Initiating Write-Off Reversals".

Note: If a payment is received for a write-off on an account, bill, or bill item, the written-off amount can be recovered and allocated to the correct account. For information on write-off reversals, see "Configuring Write-Offs and Write-Off Reversals".

About Account Groups

Accounts can be organized into groups for billing purposes or to represent the relationships between the accounts graphically in Customer Center. There are several kinds of account groups:

- A hierarchical group has a parent account and any number of child accounts and other hierarchical groups. Hierarchical account groups are created for two reasons:
  - To roll the charges of a child account up to the parent account during billing.
  - Solely to display relationships between accounts. In such cases, the parent account pays *none* of a child account's charges.

  A child account can belong to only one parent account.

  For more information, see "About Hierarchical Account Groups", "Creating Hierarchical Groups", and "Managing Hierarchical Groups".

  Note: Sponsor groups are supported for backward compatibility. If you have not set up sponsorship, use resource sharing groups instead.

- A resource sharing group consists of an owner account or service and one or more member accounts or services. Resource sharing groups are created so that discounts and charges can be shared between accounts.

  For more information, see "About Resource Sharing Groups".

- A sponsored top-up group consists of an owner account and one or more member accounts. The owner account can top up a specified balance in each member account. For more information, see "About Topping Up Accounts" in *BRM Configuring and Collecting Payments*.
- A sponsor group consists of an owner account and one or more member accounts. Sponsor groups are created so that bills and payments can be split between accounts.

  For more information, see "About Sponsor Groups".

In all types of account groups, the bills and payments belong to the account's bill units. For more information, see "About Bill Units and Account Groups".

About Bill Units and Account Groups

Accounts can have one or more bill units. Each bill unit stores billing information and tracks the charges for a particular bill. When accounts are set up with group relationships, the bill units in the accounts, not the accounts themselves, are the paying and nonpaying entities.

In hierarchy, sponsorship, and resource sharing groups, bill units have an additional internal group structure:

- A *hierarchical group* has a parent bill unit and any number of child bill units and other hierarchical bill unit groups. The charges of a child bill unit are rolled up to the parent bill unit during billing. The parent bill unit is a paying (nonsubordinate) bill unit and the child bill unit is a nonpaying (subordinate) bill unit.

If the account hierarchy is set up solely to display relationships between accounts, the bill unit in the parent account pays *none* of the charges in a child account's bill unit. Both the parent and child account's bill units are paying (nonsubordinate) bill units.

A child bill unit can belong to only one parent bill unit. The bill unit hierarchy must coincide with the account group hierarchy.

For more information, see "About Hierarchical Bill Units", "Creating Hierarchical Groups", and "Managing Hierarchical Groups".

- A *sponsor group* consists of an owner (sponsor) bill unit and one or more member (sponsored) bill units. The owner bill unit receives sponsored charges from member bill units immediately after the events generating the sponsored charges occur. An owner bill unit can pay a *portion* of a member bill unit's charges. A member bill unit can be sponsored by multiple owner bill units in multiple accounts.

  For more information, see "About Sponsor Groups".

- A *resource sharing group* consists of a group owner and one or more members. The group owner's account has an owning balance group. The owning balance group bears the financial impact of the sharing group, and the bill unit for this balance group pays for the member bill unit's charges.

  Member accounts can have more than one bill unit. If a member account has multiple bill units, the bill unit that benefits from resource sharing is the one that has the balance group whose service has been chosen for resource sharing.

  For more information, see About Resource Sharing Groups.

For a detailed comparison of the hierarchy, sponsorship, and resource sharing features, see "Comparing Hierarchy, Resource Sharing, and Sponsorship".

To understand the financial implications of creating and managing account groups, see "Hierarchy Balance Impacts" and "Sponsorship Balance Impacts".

Comparing Hierarchy, Resource Sharing, and Sponsorship

Hierarchical, resource sharing, and sponsor groups enable customers to pay other customer's bills, but they do so in different ways. These are the main differences:

Relationships

- **Hierarchy**: Represents parent/child relationships between accounts and account bill units.
- **Resource sharing**: Represents owner/member relationships in which the owner assumes certain charges for the member or shares resources with the members. Discount sharing is based on discounts purchased by the owner as part of a product rate plans. Charge sharing is based on chargeshares in BRM.
- **Sponsorship**: Represents owner/member relationships in which the owner sponsors product rate plans for one or more member bill units.

Bill Units and Balance Groups

- **Hierarchy**: Parent and child accounts have either paying nonsubordinate bill units or nonpaying subordinate bill units.

- **Resource sharing**: Sharing group owner accounts have owning balance groups that are part of bill units. For charge sharing, charges for eligible member events impact the owning balance group and associated bill unit. For discount sharing, shared discounts impact the member's balance groups by either increasing non-currency resources or reducing the currency impact of an event.
- **Sponsorship**: Sponsor owner accounts have sponsor owner bill units. Member accounts have sponsored member bill units. The sponsor owner bill unit pays for some charges generated by a sponsored member bill unit.

  Sponsor owner accounts can themselves be part of a sponsorship and thus have member bill units that are sponsored by a different owner. Member accounts can be sponsor group owners and thus have owner bill units that sponsor charges for a different set of members. However, the sponsor group owner cannot be a member of a sponsor group owned by one of its members.

Product and Service Ownership

- **Hierarchy**: Nonpaying (subordinate) bill units of the same parent bill unit do not have to own the same products.
- **Resource sharing**: An owner account does not need to have the same products or services as the members.

  The owner bill units can provide resource sharing to group members even if, at group creation time, some members do not own the services for which sharing is provided. In this case, only members who currently own the shared services benefit from discount or charge sharing. Members who do not own the service when sharing is set up can participate in sharing if they purchase that service in the future.
- **Sponsorship**: For an owner (sponsor) bill unit to incur charges for its member (sponsored) bill units, all members of the owner's bill unit must own the products sponsored by that group. (Members can own different additional products.)

Product Guidelines

- **Hierarchy**: A parent bill unit can pay for any product.
- **Resource sharing**: An owner can pay for a member's charges in accordance with a **ChargeShare** selected when the sharing group is created. An owner can share any discount included in the plans they purchase.
- **Sponsorship**: To be paid for by a sponsor owner bill unit, a product must be set up in Pricing Center as sponsorable.

Number of Owners and Members

- **Hierarchy**: A child account and bill unit can belong to only one parent.
- **Resource sharing**: Multiple owners can provide discount or charge sharing for the same service, and a member account can participate in multiple sharing groups for a service. Thus, a member bill unit can share charges with multiple owner bill units or receive discounts from multiple owners.
- **Sponsorship**: A member account can be sponsored by multiple sponsor group owner accounts. Likewise, a member bill unit can be sponsored by multiple sponsor owner bill units.

  Each owner should sponsor a different product. See "One Sponsor per Product" in *BRM Setting Up Pricing and Rating*.

What the Parent or Owner Pays For

- **Hierarchy**: A parent bill unit pays all the charges for its nonpaying subordinate bill units. The parent cannot pay for only some of a subordinate's charges.

  Bill items are created for nonpaying (subordinate) bill units, but the amounts in them are rolled up at billing to the paying bill unit's bill.

  Exception: If a subordinate bill unit is also a member of a sponsor group, its sponsored charges are paid by the sponsor group's owner bill unit.

- **Resource sharing**: An owner bill unit pays only the member charges that are defined as shared when the charge sharing group is created.

  A charge sharing group can include some or all chargeshares in the database. The charges go directly to the bill items of the sharing group owner's bill unit.

- **Sponsorship**: A sponsor owner bill unit pays only its members' sponsored charges.

  You can sponsor some or all rate plans in a product. The sponsored charges go directly to the sponsor owner bill unit's bill items.

Multiple Groups and Multiple Bill Units

- **Hierarchy**: If a child account has multiple bill units, a parent account can pay the charges for one, several, or all of the child account's bill units.
- **Resource sharing**: If a member account has a service that participates in multiple sharing groups from different owner accounts, the member's charges are distributed among the groups' owners and the member benefits from each owner's shared discounts. BRM distributes the charges among the owner bill units and applies discounts based on priorities defined when the member joins the resource sharing groups. The member account pays any charges that are not eligible for sharing.
- **Sponsorship**: If a member account belongs to multiple sponsor groups that are owned by different accounts and that sponsor different products, the member's charges are split accordingly among the groups' owners. The member account pays all its own unsponsored charges.

Collecting Payment Information

- **Hierarchy**: During account creation, payment information is not collected for child accounts' nonpaying bill units. Their payment method is **Paid by parent account**.
- **Resource sharing**: During charge sharing group creation, the payment method of the owning balance group is set as the payment method for any shared charges. The members' payment method has no bearing on the resource sharing so members can use any payment method.

  Discounts do not generate bills, they only change the amount of a bill. Therefore, payment method is not a factor for discount sharing groups.

- **Sponsorship**: During account creation, payment information is collected for all member accounts' bill units that do not participate in the sponsorship (paying, subordinate bill units). Members can use any payment method.

Tracking Balance Impacts

- **Hierarchy**: Nonpaying (subordinate) bill units track and display their own balance impacts in real time. When you bill accounts in a hierarchical group, balance impacts of nonpaying bill units are rolled up to the parent bill unit, and the parent account is billed for them.

- **Resource sharing**: Member bill units do not track or display the balance impacts of shared charges. The owning balance group tracks the balance impacts of usage resulting from non-currency shared discounts; for example, free minutes.
- **Sponsorship**: Members of sponsored bill units do not track or display the balance impacts of sponsored charges.

Resources

- **Hierarchy**: Hierarchical groups deal only with billing, so only currency resources are affected.
- **Resource sharing**: Sharing groups handle all types of resources. For example, you can apply an owner's discount for free minutes to each member or apply a 15% discount on a member's monthly fee.
- **Sponsorship**: Sponsor groups handle all types of resources. For example, you can apply a non-currency credit to a sponsor owner bill unit from each sponsor member bill unit.

Hierarchy Balance Impacts

The parent-child relationship has no financial impact unless the child bill unit is a nonpaying bill unit. Nonpaying bill units in child accounts maintain all their charges until billing, when the charges are rolled up to the paying bill unit of the parent account, who is then responsible for the bill.

There is no financial impact when you move an independent account or bill unit into a hierarchy and make it a child. If you make the bill unit a nonpaying (subordinate) bill unit, then you choose whether to bill the subordinate bill unit for bill-in-progress charges or to transfer the charges to the new parent, nonsubordinate bill unit.

When you move a nonpaying bill unit from one paying parent to another, you can determine which parent bill unit will pay the bill-in-progress charges. By default, the bill-in-progress charges transfer to the new parent bill unit.

Before you can move a nonpaying bill unit out of a hierarchy, you must change it to a paying bill unit and indicate whether the bill unit or its former parent will pay any bill-in-progress charges.

Resource Sharing Group Balance Impacts

When a member account's service is added to a resource sharing group, a financial relationship is created between the sharing group owner account and the member account.

- **Charge sharing**: If the group owner account is active and the member service has events eligible for charge sharing, the shared charges impact the balance of the group owner account rather than the member account.
- **Discount sharing**: If the group owner account is active and the member account has services eligible for discount sharing, the shared discounts reduce the amount of money a member owes by applying the owner's discounts to the balance of the member account before finalizing the member's charges.

The member account stops benefiting from the resource sharing group in the following situations:

- The owner account is inactive or closed.
- The resource sharing group is deleted from the owner account.
- The member account's service is removed from the sharing group.

- Events generated by a member fall outside of the validity period for the shared discount or charge.
- The discounts or chargeshares that were included in the sharing group are deleted through Pricing Center.
- For discount sharing groups, the non-currency resources that were offered as part of the discount have been depleted in the owner account; for example, a monthly discount of free minutes that has already been consumed by the owner or other members. In this case, the member still benefits from currency-type shared discounts such as a 10% reduction on bills for email usage or a $15 discount on overseas calls.

Sponsorship Balance Impacts

When an account is added to a sponsor group and sponsored products are purchased for the account, a financial relationship is created between the sponsor owner bill unit and the member bill unit. If sponsorship is guaranteed and the sponsor owner bill unit remains active, sponsored charges affect the balance of the sponsor owner bill unit rather than the member bill unit.

The member bill unit, rather than the sponsor owner bill unit, receives sponsored charges in the following situations:

- Sponsorship is not guaranteed and the sponsor bill unit reaches its credit limit.
- The sponsor group owner account or bill unit is inactivated.
- The sponsor group owner account or bill unit is closed, the sponsor group is deleted rather than reassigned, and the sponsored products are not canceled.
- The member account or bill unit is removed from the group.
- The sponsor group is deleted from the sponsor group owner account or bill unit.

Managing Balance Groups with Your Custom Application

For information about how BRM uses balance groups, see "About Balance Groups".

To manage balance groups, use the following opcodes:

- PCM_OP_CUST_SET_BAL_GRP

  See "Creating Balance Groups".
- PCM_OP_CUST_MODIFY_CUSTOMER

  See "Moving a Balance Group from One Bill Unit to Another".
- PCM_OP_CUST_DELETE_BAL_GRP

  See "Deleting a Balance Group".

To modify a sub-balance, see "Modifying a Sub-balance".

Creating Balance Groups

PCM_OP_CUST_SET_BAL_GRP is a wrapper opcode that performs all necessary tasks to set up the **/balance_group** object and create a link to the customer account.

The PCM_OP_CUST_COMMIT_CUSTOMER opcode calls the PCM_OP_CUST_SET_BAL_GROUP opcode while creating a customer account and while modifying customer account information.

If the PIN_FLD_SERVICE_OBJ field in the input flist has a NULL value, the PCM_OP_CUST_SET_BAL_GRP opcode creates a balance group associated with the account. If the PIN_FLD_SERVICE_OBJ field specifies a service object POID, a balance group is created and associated with the service.

PCM_OP_CUST_SET_BAL_GROUP calls the following opcodes:

- PCM_OP_BILL_SET_LIMIT_AND_CR to set the credit limit in the balance group.
- PCM_OP_CUST_CREATE_BAL_GRP to create the **/balance_group** object.
- PCM_OP_CUST_MODIFY_BAL_GRP to modify an existing **/balance_group** object.

  Based on which field PCM_OP_CUST_MODIFY_BAL_GRP is modifying, the following fields are either mandatory or optional:

  – PIN_FLD_NAME

  – PIN_FLD_BILLINFO_OBJ

  While modifying the balance group, PCM_OP_CUST_SET_BAL_GROUP checks if there are any pending bill items. If there are pending bill items and if the billing information has changed, PCM_OP_CUST_SET_BAL_GROUP changes the billing information with the new billing information.

If successful, the PCM_OP_CUST_SET_BAL_GROUP output flist contains the POID of the **/balance_group** object that is created or modified.

Specifying the Default Balance Group of a Bill Unit

To specify your custom algorithm to select the default balance group of a bill unit, use the PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC policy opcode.

By default, this policy opcode is an empty hook provided for customization. You can add your own custom algorithm to override the BRM algorithm for selecting the default balance group of a bill unit. See "About the Default Balance Group of a Bill Unit".

Important: By overriding the BRM algorithm, you will change the system behavior that could impact other functionality. For example, if you choose to use only the object creation time or the object POID as the selection criterion, multiple balance group objects might qualify. You should have a good understanding of the default implementation before you customize this policy opcode.

Specifying the Default Service of the Default Balance Group

To specify your custom algorithm to select the default service of the default balance group, use the PCM_OP_BAL_POL_GET_BAL_GRP_AND_SVC policy opcode.

By default, this policy opcode is an empty hook provided for customization. You can add your own custom algorithm to override the BRM algorithm for selecting the default service of the default balance group. See "About the Default Service of the Default Balance Group".

Important: By overriding the BRM algorithm, you will change the system behavior that could impact other functionality. For example, if you choose to use only the object creation time or the object POID as the selection criterion, multiple service objects might qualify. You should have a good understanding of the default implementation before you customize this policy opcode.

Moving a Balance Group from One Bill Unit to Another

To move a balance group from one bill unit (**/billinfo** object) to another in the same account, use the PCM_OP_CUST_MODIFY_CUSTOMER opcode.

Important: You cannot move account-level balance groups or default balance groups to a different **/billinfo** object.

Moving a balance group to a different **/billinfo** object means that any new charges for the services in the balance group will be applied to the new **/billinfo** object.

For example, an account has two **/billinfo** objects. One **/billinfo** object tracks charges for services that are invoiced. The other tracks charges for services paid by credit card. The customer decides to pay for all services by credit card. The balance group for invoiced services is moved to the **/billinfo** object with the credit card payment method. Any new charges for these services are applied to the new **/billinfo** object and are charged to the credit card. Existing charges for these services that occurred before the balance group was moved are associated with the old **/billinfo** object and are invoiced.

You cannot move a balance group to another **/billinfo** object if the balance group's **/billinfo** object has unallocated payments or adjustments, open refunds, or unresolved disputes. All disputes must be settled, refunds paid, and payments and adjustments allocated before the balance group can be moved.

A **/billinfo** object must have at least one balance group. When a **/billinfo** object has only one balance group and that balance group is moved to another **/billinfo** object, PCM_OP_CUST_MODIFY_CUSTOMER automatically creates a new balance group not associated with any service for the **/billinfo** object from which the balance group was moved.

Important: You should never move a balance group to a **/billinfo** object in a different account. To have a different account be responsible for the charges in a balance group, you must create an account and **/billinfo** hierarchy.

Deleting a Balance Group

To delete a balance group, use the PCM_OP_CUST_DELETE_BAL_GRP opcode.

Modifying a Sub-balance

You can use Customer Center to change validity dates and apply debits and credits to any sub-balance in an account-level balance group. For more information, see the Customer Center Help.

Use the following opcodes to modify sub-balances associated with additional **/billinfo** objects. Specify the **/balance_group** object that contains the sub-balance to modify on the input flist:

- PCM_OP_BAL_CHANGE_VALIDITY changes a sub-balance validity period.

  For more information, see "Modifying the Sub-balance Validity Period".
- PCM_OP_BILL_DEBIT debits or credits a non-currency sub-balance.
- PCM_OP_AR_ACCOUNT_ADJUSTMENT debits or credits a currency sub-balance. Use this opcode to adjust the balance for a prepaid currency resource.
- PCM_OP_BILL_TRANSFER_BALANCE transfers a positive balance from one **/balance_group** object to another when the balance groups are specified on the input flist. The source and destination balance groups can be in different accounts.
- PCM_OP_BILL_SET_LIMIT_AND_CR sets the credit limit for currency and non-currency resources in a sub-balance.

Modifying the Sub-balance Validity Period

Resource validity periods define when a granted resource is available for consumption.

Customer Center calls the PCM_OP_BAL_CHANGE_VALIDITY opcode to update the valid dates for sub-balances in **/balance_group** objects. In turn, PCM_OP_BAL_CHANGE_VALIDITY calls the PCM_OP_ACT_USAGE opcode to generate **/event/billing/sub_bal_validity** events.

To change the validity period of resource balances, specify the following fields in the PIN_FLD_SUB_BALANCES array on the input flist:

- If you are changing the validity period of more than one resource:
  - Specify the resource ID in PIN_FLD_RESOURCE_ID.
  - Specify the sub-balance array index in PIN_FLD_ELEMENT_ID.
- If you are setting absolute dates:
  - Specify the new start time in PIN_FLD_VALID_FROM.
  - Specify the new end time in PIN_FLD_VALID_TO.

Transferring Amounts between Items

Transfers are usually made to correct payments that were allocated incorrectly. Transfers are also used to allocate an account-level debit adjustment for any account and to allocate an account adjustment for an account that pays by credit card. In Customer Center, a CSR can transfer amounts between bill items. Transfers can also be made from payment items and account adjustment items to bill items.

How Resources Are Transferred

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

Use the following opcodes for transfers:

- PCM_OP_BILL_ITEM_TRANSFER

  See "Transferring Resources between Items".

- PCM_OP_BILL_TRANSFER_BALANCE

  See "Transferring Resources between Balance Groups".
- PCM_OP_BILL_POL_VALID_TRANSFER

  See "Customizing Payment Transfer Validation".

Transferring Resources between Items

To transfer resources between items, use the PCM_OP_BILL_ITEM_TRANSFER opcode.

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

PCM_OP_BILL_ITEM_TRANSFER is called by these opcodes:

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

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

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

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

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

PCM_OP_BILL_ITEM_TRANSFER:

- Reads the target item.

- Changes the target item's fields. The fields changed depend on the source item. For example, if the source item is an adjustment, the PIN_FLD_ADJUSTED field of the target item is changed.
- Changes the source item fields as follows:

  PIN_FLD_DUE -= PIN_FLD_AMOUNT

  PIN_FLD_TRANSFERED += PIN_FLD_AMOUNT
- If PCM_OP_BILL_ITEM_TRANSFER was called as part of settling a dispute, checks the PIN_FLD_DISPUTED field in the flist to determine the dispute amount.

  The PIN_FLD_DISPUTED field is only populated in the flist if PCM_OP_BILL_ITEM_TRANSFER is called by an item-level and event-level settlement, and it is used to prevent misdirection of settlement resources. It ensures proper application of the settlement in cases where a single item may be disputed by both an item-level dispute and an event-level dispute.

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

  Using this information, PCM_OP_BILL_ITEM_TRANSFER populates the settled amount in the original item's PIN_FLD_ADJUSTED field to record the balance impact of the settlement. If any of the disputed amount was denied in the settlement, the opcode also increases the PIN_FLD_DUE field in the original item by the denied amount. These activities are summarized as follows:

  PIN_FLD_DISPUTED -= PIN_FLD_DISPUTED

  PIN_FLD_ADJUSTED += PIN_FLD_AMOUNT

  PIN_FLD_DUE += (PIN_FLD_DISPUTED - PIN_FLD_AMOUNT)
- Writes the **/event/billing/item/transfer** event, which includes the following fields in addition to those it inherits:
  - PIN_FLD_BUFFER: List of all items or bills affected.
  - PIN_FLD_AMOUNT: Amount of the transfer.
  - PIN_FLD_AR_BILL_OBJECT: A/R bill object of the items in the transfer.
  - PIN_FLD_SESSION_OBJ: The associated A/R event or batch event in which the A/R action took place (payment batch event, refund batch event, or reversal batch event.
- If (PIN_FLD_DUE == PIN_FLD_DISPUTED == 0) marks the status PIN_ITEM_STATUS_CLOSED.
- Writes the modified source and target items.

Transferring Resources between Balance Groups

To transfer resources between balance groups, use the PCM_OP_BILL_TRANSFER_BALANCE opcode.

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

By default, PCM_OP_BILL_TRANSFER_BALANCE transfers a resource from a credit balance to another balance by debiting the source balance and crediting the target balance with the PIN_FLD_AMOUNT specified in the opcode's input flist. For example:

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

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

- Balance group A has a debit balance of $10 (represented as **+10**).
- Balance group B has a credit balance of $2 (**-2**).
- If PIN_FLD_AMOUNT is **30** and PIN_FLD_VERIFY_BALANCE is **PIN_BOOLEAN_FALSE**, PCM_OP_BILL_TRANSFER_BALANCE transfers $30 from balance group A to balance group B with these results:
  - Balance group A now has a debit balance of $40 (**+40**).
  - Balance group B now has a credit balance of $32 (**-32**).

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

For more information, see "About Transferring Sponsored Top-Ups from Debit Balances" in *BRM Configuring and Collecting Payments*.

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

Table 1–5 lists the valid entries for the PIN_FLD_RESULT field:

Table 1–5 PIN_FLD_RESULT Values

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

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

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

Customizing Payment Transfer Validation

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

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

Transferring Services between Balance Groups by Using Custom Client Applications

CSRs can use a custom client application to transfer a service from one balance group to another within the same account or in a different account. When this occurs, the custom client application calls the PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER opcode, which automatically performs all tasks necessary to transfer the service.

The opcode stores data about a service's old and new balance group and the timestamp of when the transfer occurred in the **/service** object's PIN_FLD_TRANSFER_LIST array. When BRM applies balance impacts or retrieves balance group information, it automatically checks the **/service** object's PIN_FLD_TRANSFER_LIST array to determine which balance group it should use based on the event timestamp. If the array does not list a balance group, the opcode automatically uses the service's default balance group.

For more information, see "About Transferring Services between Balance Groups".

To set up your system to transfer services between balance groups, perform these tasks:

- Customize your client application to call PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER.

  See "Transferring Services in Custom Client Applications".
- Configure BRM to synchronize the balance group transfer data with Pipeline Manager.

  See "Synchronizing Balance Group Transfer Data with Pipeline Manager".

Transferring Services in Custom Client Applications

To transfer a service from one balance group to another, use the PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER opcode.

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

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

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

PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER performs the following tasks:

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

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

   ---

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

   If it is, all member services must be transferred and reassigned. The opcode performs the following validations:
   - If the transfer is across accounts, that the member services were not passed in the input flist, that the subscription service and member services are assigned to the same balance group, and that the services passed in the input flist all point to PIN_FLD_FROM_BALANCE_GROUP.

- If the transfer is within the same account, that the services passed in the input flist all point to PIN_FLD_FROM_BALANCE_GROUP.

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

   If the array was passed in, a new deal was purchased and all of the service's existing deals must be canceled. The opcode performs the following:

   a. Calls the PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS opcode to retrieve the account's hierarchical relationships of deals, products, discounts, and services.

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

7. Calls the PCM_OP_CUST_SET_BAL_GRP opcode.

   The PCM_OP_CUST_SET_BAL_GRP opcode performs one of the following:

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

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

8. Locks the balance group that the service is transferring to.

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

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

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

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

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

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

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

13. Generates the **/event/notification/service_balgrp_transfer/data** notification event.

    By default, this triggers a call to the PCM_OP_IFW_SYNC_PUBLISH_EVENT opcode.

14. Generates the **/event/notification/service_balgrp_transfer/end** notification event.

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

Synchronizing Balance Group Transfer Data with Pipeline Manager

If your system uses both real-time rating and batch rating, you must configure BRM to notify Pipeline Manager when:

- A service transfers to a different balance group.
- A balance group transfers to a different bill unit.

BRM notifies Pipeline Manager through the **ServiceBalanceGroupTransfer** business event.

To configure BRM to notify Pipeline Manager when a service transfers to a different balance group, install the Account Synchronization DM and configure it to publish the **ServiceBalanceGroupTransfer** business event to the account synchronization queue. For detailed installation and configuration instructions, see "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.

When configuring the Account Synchronization DM:

- Make sure the **ServiceBalanceGroupTransfer** business event is listed in your payload configuration file under the **<PublisherDefs>** section. This business event appears in the default Account Synchronization payload configuration file (*BRM_Home*/**sys/eai_js/payloadconfig_ifw_sync.xml**).
- Make sure the event notification list maps the **/event/notification/service_balgrp_transfer/data notification** event to opcode number **3626**. This mapping appears in the default Account Synchronization event notification file (*BRM_Home*/**sys/data/config/pin_notify_ifw_sync**).
- Add the **ServiceBalanceGroupTransfer** business event to your *BRM_Home*/**sys/dm_aq/aq_queuenames** file, if the file is configured to send specific business events.

2

Using the Event Browser to Display and Adjust Events

This chapter provides an overview of the Event Browser, which is a component of Oracle Communications Billing and Revenue Management (BRM) Customer Center. You use the Event Browser to display and adjust events, including making tax and accounts receivable adjustments.

For background information, see "About Accounts Receivable".

For information about calculating taxes, see "About Calculating Taxes" in *BRM Calculating Taxes*.

Working with the Event Browser

You use Event Browser to find events. When you find the event, you can perform the following tasks:

- Review detailed information about customer accounts such as credits, debits, dialup sessions, and name changes.

  See "Finding Events".
- Summarize the impacts of events on the resource balances in an account. For example, you can see how login events during one month affected the balance of dollars and hours in a customer's account.
- Cancel the effects of events on resource balances. For example, if a customer was charged the wrong amount, you can cancel the charge by adjusting the event that generated the charge.

  See "Adjusting Events".

Note: The Event Browser shows you all data in event objects. In addition, you can look at data stored in other objects associated with an event object, such as account objects, bill objects, and service objects.

Finding Events

To find events, you use an Event Browser search template to search for events with specific characteristics. You can make your search more specific by performing an advanced search that simultaneously uses more than one search template. For example, to find a check payment event for a specific account during a certain month, use the **Check Payments** template, the **Account** template, and the **Between Dates** template.

For more information, see Event Browser Help.

Tip: To tailor your search for specific needs, you can create custom search templates. See "Creating Custom Search Templates for Event Browser" in *BRM Developer's Guide*.

Reviewing Events

The data that you see in Event Browser depends on your access privileges and permission level. For example, without appropriate access privileges, you cannot access accounts in a certain brand or account group, and without the appropriate permission, you cannot see credit card numbers. For more information about access privileges and permissions, see Event Browser Help.

After you search for events, the Results table displays the information you requested.

To see more information, you can display detailed data about an event or an object associated with an event. Detailed information appears in the Event Detail window.

Tip: Use the **Back** and **Forward** buttons in the Event Detail window to browse between details of events and objects you want to review.

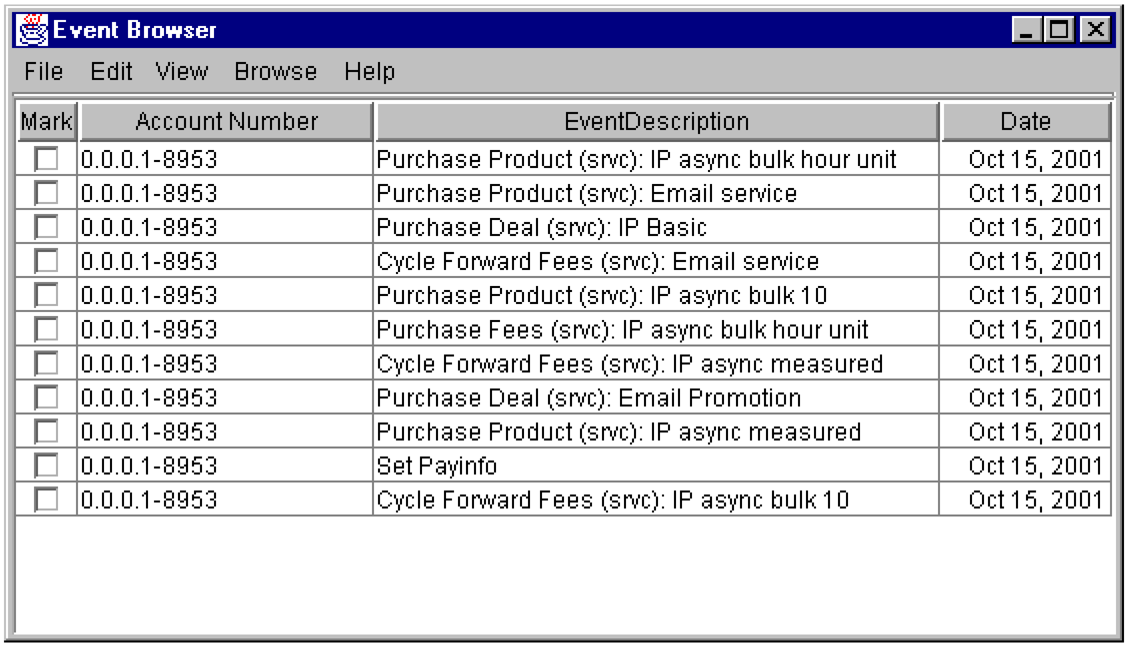
About the Results Table

After you search for an event, your search results appear in the Results table as shown in Figure 2–1. The Results table shows all the events that match your search criteria. Each matching event is displayed in a row in the table. The columns show the information stored in the events. Depending on the template you used, different columns might appear.

Note: Service passwords stored as plain text are displayed as asterisks (****) in the Event Browser.

You can customize the results table to suit your needs. See "Creating Custom Search Templates for Event Browser" in *BRM Developer's Guide*.

Figure 2–1 Event Browser



Event Browser

Mark	Account Number	EventDescription	Date
☐	0.0.0.1-8953	Purchase Product (srvc): IP async bulk hour unit	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Product (srvc): Email service	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Deal (srvc): IP Basic	Oct 15, 2001
☐	0.0.0.1-8953	Cycle Forward Fees (srvc): Email service	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Product (srvc): IP async bulk 10	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Fees (srvc): IP async bulk hour unit	Oct 15, 2001
☐	0.0.0.1-8953	Cycle Forward Fees (srvc): IP async measured	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Deal (srvc): Email Promotion	Oct 15, 2001
☐	0.0.0.1-8953	Purchase Product (srvc): IP async measured	Oct 15, 2001
☐	0.0.0.1-8953	Set Payinfo	Oct 15, 2001
☐	0.0.0.1-8953	Cycle Forward Fees (srvc): IP async bulk 10	Oct 15, 2001

Depending on how you define your search criteria, the Results table could display one event or many events. If no events match your search criteria, you will see a message and the Event Browser displays a blank table.

Events are sorted in the Results table by the column your company specified when defining the search template properties. You can sort the **Event Description** and **Date** columns; click the column head to sort in descending order, press SHIFT and click the column head to sort in ascending order.

For more information, see Event Browser Help.

Time Zone Indication in Event Browser

Customer Center can display the event time based on one of the following:

- The time zone where the event occurred.
- The time zone where the server was located.
- The time zone where the CSR is located.

Table 2–1 lists how the Event Browser Event Detail panel indicates the time zone with one, two, or three asterisks (*):

Table 2–1 Time Zone Representation in Event Browser

Number of Asterisks (*)	Represents the Time Zone...
One (*)	Where the event occurred.
Two (**)	Where the server is located.
Three (***)	Where the CSR is located. This is the default.

For the time displayed in the Start Date, End Date, and Effective Date fields, Event Browser does the following:

1. Looks for the event time zone in the event TIMEZONE__ID field.

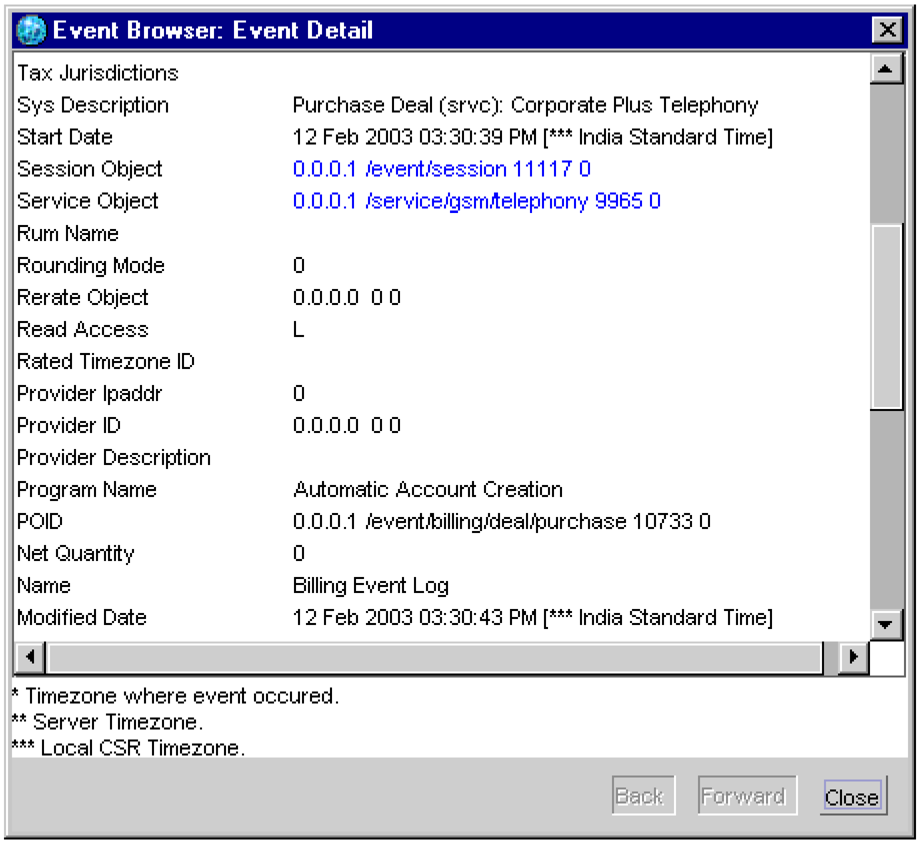
2. If there is no entry in the TIMEZONE__ID field, Event Browser gets the server time zone from the **Infranet.properties** file **infranet.server.timezone** entry.
3. If this entry is not configured, Event Browser uses the CSR time zone.

For all other fields that show the event time, Event Browser does the following:

1. Looks for the server time zone in the **Infranet.properties** file **infranet.server.timezone** entry.
2. If this entry is not configured, Event Browser uses the CSR time zone.

Figure 2–2 shows the Event Detail panel with the captions:

Figure 2–2 Event Detail Panel



Tax Jurisdictions	
Sys Description	Purchase Deal (srvc): Corporate Plus Telephony
Start Date	12 Feb 2003 03:30:39 PM [*** India Standard Time]
Session Object	0.0.0.1 /event/session 11117 0
Service Object	0.0.0.1 /service/gsm/telephony 9965 0
Rum Name	
Rounding Mode	0
Rerate Object	0.0.0.0 0 0
Read Access	L
Rated Timezone ID	
Provider Ipaddr	0
Provider ID	0.0.0.0 0 0
Provider Description	
Program Name	Automatic Account Creation
POID	0.0.0.1 /event/billing/deal/purchase 10733 0
Net Quantity	0
Name	Billing Event Log
Modified Date	12 Feb 2003 03:30:43 PM [*** India Standard Time]

* Timezone where event occured.
** Server Timezone.
*** Local CSR Timezone.

Setting the Time Zone Displayed in Event Browser

To set the time zone to be displayed in the Event Detail panel:

1. Open the **Infranet.properties** file in the *BRM_Home*/**EventBrowser** directory:
2. Edit the **infranet.server.timezone** entry:

   ```
   infranet.server.timezone=timezone
   ```

   where *timezone* represents the three-letter time zone code such as PST.
3. Save and close the file.

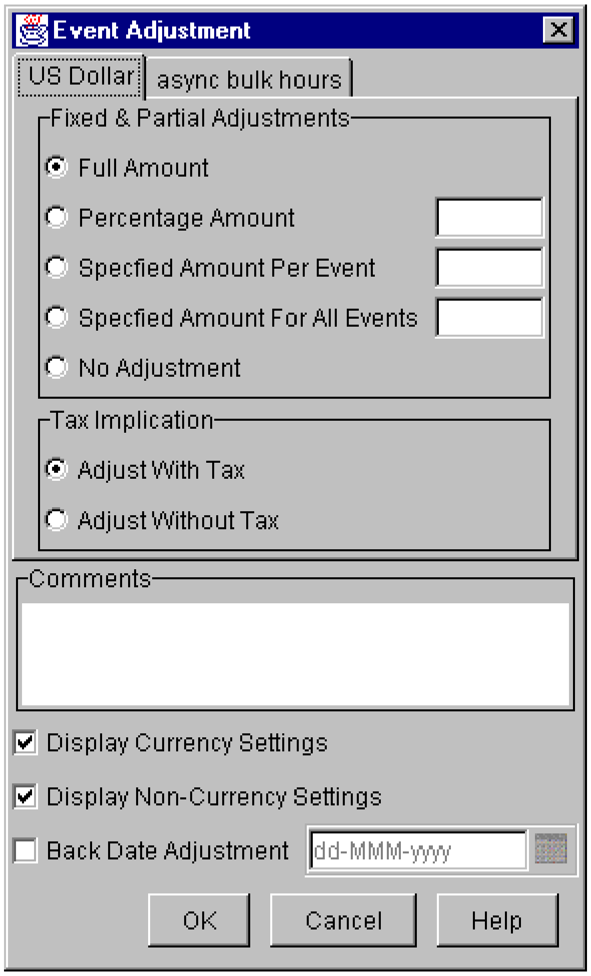
Adjusting Events

You adjust events to cancel the effect of the events on resource balances and general ledger entries. An event adjustment does not cancel the event itself. For example, if you adjust an event generated when a password is changed, the event adjustment does not cancel the password change, only the cost of the change.

Note: You can adjust events for more than one account at a time.

Figure 2–3 shows the options you can choose when adjusting events:

Figure 2–3 Event Adjustment



Event Adjustment

US Dollar | async bulk hours

Fixed & Partial Adjustments
- [✓] Full Amount
- [ ] Percentage Amount
- [ ] Specfied Amount Per Event
- [ ] Specfied Amount For All Events
- [ ] No Adjustment

Tax Implication
- [✓] Adjust With Tax
- [ ] Adjust Without Tax

Comments

- [✓] Display Currency Settings
- [✓] Display Non-Currency Settings
- [ ] Back Date Adjustment dd-MMM-yyyy

If you do a partial event adjustment, you must also specify whether to adjust a percentage of each marked event, a specific amount for each event, or a single amount for the group of events being adjusted at the same time.

If you choose to adjust accounts receivable or events, you must also set the options in the Event Adjustment dialog box.

Note: You won't be able to adjust an event unless you have the read/write or write-only permission level for the **/accounttool/araccess** permission. For more information about permissions, see Event Browser Help.

Caution:

- Adjusting an event affects the balance in the customer's account. You should adjust an event only in strict compliance with your company's guidelines.
- Be sure to confirm the settings on every resource tab in the Event Adjustment dialog box before clicking **OK**. If you do not want the adjustment to affect a particular resource, be sure the **No Adjustment** button is selected on the tab for that resource.

By default, you can adjust the following events:

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

If you create custom events that you must adjust or if you must adjust additional events, you must add additional event types to this configuration object. See "Adding Fields to /config Objects" in *BRM Developer's Guide*.

Printing Events

You can print information in the event object, but you cannot print information stored in linked objects, such as the associated account or bill objects.

For information, see Event Browser Help.

Exporting an Event Description

You can save event descriptions by exporting them to a text file. You can export information in the event object, but you cannot export information stored in linked objects, for example, the associated account or bill objects.

For information, see Event Browser Help.

Customizing the Event Browser

You can customize the Event Browser by adding custom search templates and defining custom results tables. For more information, see "Creating Custom Search Templates for Event Browser" in *BRM Developer's Guide*.

3

Creating and Managing Hierarchical Account Groups

This chapter describes how to create and manage hierarchical account groups in your Oracle Communications Billing and Revenue Management (BRM) system.

For information about different types of groups, see "About Account Groups".

Before working with account and bill unit groups, you should be familiar with BRM rating and billing. See "About Creating a Price List" in *BRM Setting Up Pricing and Rating* and "About Billing" in *BRM Configuring and Running Billing*.

About Hierarchical Account Groups

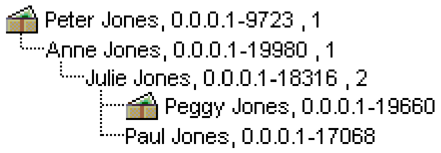
A hierarchical account group is a set of accounts organized according to their positions in relation to each other. The relationships among accounts in a hierarchical group are similar to parent-child relationships. A hierarchical group is headed by a parent account with child accounts beneath it. At each level above the bottom of the hierarchy, the child accounts themselves can be parent accounts.

You can set up account groups for billing purposes. For example, a corporate customer might have several accounts for corporate employees, but the corporation itself pays all the employees' bills. In this case, the corporation's account is the parent account with the paying bill unit, and the employees' accounts are child accounts with nonpaying bill units.

An account's position in a hierarchy does not necessarily indicate whether it pays its own bills. Any account, either a parent or child, can have a paying bill unit or a nonpaying bill unit.

Note: The top-level parent account must have a paying bill unit. For example, in Figure 3–1, Peter Jones must have a paying bill unit. (In Customer Center, the wallet icon identifies the account with a paying bill unit.)

Figure 3–1 Top-Level Parent Account with Paying Bill Unit



```
Peter Jones, 0.0.0.1-9723 , 1
└Anne Jones, 0.0.0.1-19980 , 1
  └Julie Jones, 0.0.0.1-18316 , 2
    ├Peggy Jones, 0.0.0.1-19660
    └Paul Jones, 0.0.0.1-17068
```

For more information about paying and nonpaying bill units in accounts, see "A/R and Hierarchical Account Groups".

For more information about paying and nonpaying bill unit hierarchies, see "About Hierarchical Bill Units".

How Account Status Changes Affect Hierarchies

By default, changing the status of the parent account in a hierarchical account group changes the status of all subordinate bill units in the group. For more information, see "How Bill Unit Status Changes Affect Hierarchies".

Brand Requirements of Hierarchies

If your company supports Branded service management, parent and child accounts must belong to the same brand.

Performance Impact of Account Hierarchies

To maintain reliable data consistency, many operations lock an account at the beginning of the transaction. Therefore, in an account hierarchy, many of the associated accounts are also locked. While this provides reliable data consistency, it can cause a lot of serialization which decreases the throughput of the system.

If, with account hierarchies, this problem affects your system, you may choose to lock specific balance groups instead of the whole account.

Important: Balance Group locking may enable separate contexts to attempt to lock the same object causing a system halt. Whenever balance group locking is used, every feature that uses balance group locking should be examined for overlap.

For more information on Balance Group Locking feature, see "Locking Specific Objects" in *BRM Developer's Guide*.

About Hierarchical Bill Units

Hierarchical bill units are a set of bill units organized according to their positions in relation to each other and the accounts to which they belong. A hierarchical bill unit group is headed by a parent bill unit with child bill units beneath it. A parent bill unit belongs to a parent account and a child bill unit belongs to a child account. At each level above the bottom of the hierarchy, the child bill units themselves can be parent bill units.

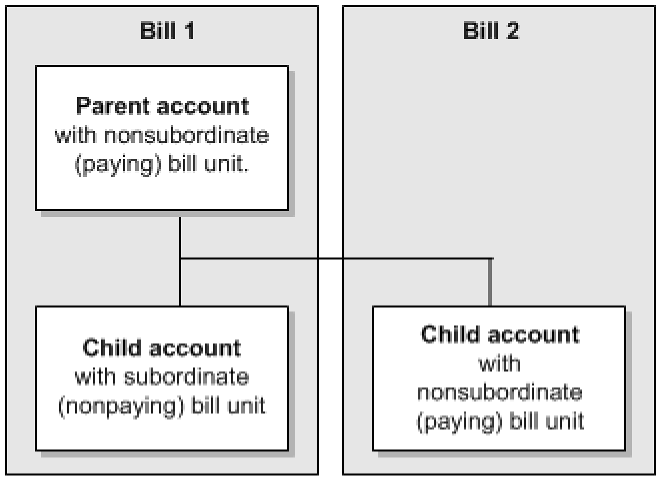
For billing purposes, child accounts can have nonpaying bill units or paying bill units:

- A nonpaying bill unit's bill is paid by the paying bill unit in the parent account.
- A paying bill unit in a child account pays its own bill.

A bill unit's position in a hierarchy does not necessarily indicate whether it pays its own bills. Any bill unit, either a parent or child, can be a paying bill unit or a nonpaying bill unit.

Figure 3–2 shows a simple hierarchical group. There are three bill units, one in each account, but because the paying bill unit in the parent account pays the bill for the nonpaying bill unit in the child account, there are only two bills for the three accounts:

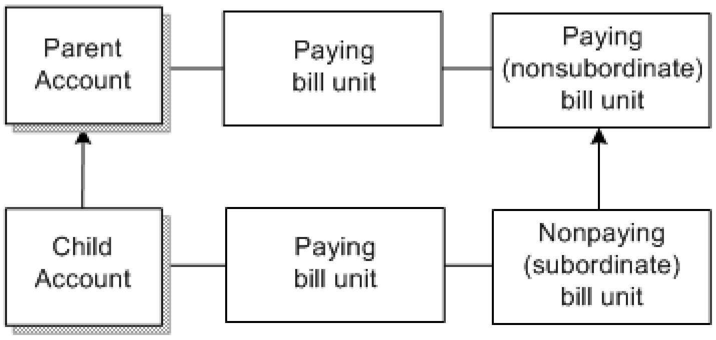
Figure 3–2 Simple Hierarchical Group


Bill 1
Bill 2
Parent account with nonsubordinate (paying) bill unit.
Child account with subordinate (nonpaying) bill unit
Child account with nonsubordinate (paying) bill unit

When accounts have multiple bill units, the bill unit hierarchy becomes more complex. A child account can have both nonpaying bill units and paying bill units. The parent account is not required to pay all of the child account's bills.

Figure 3–3 shows two accounts with two bill units each, but only one of the child account's bill units is subordinate to the parent account:

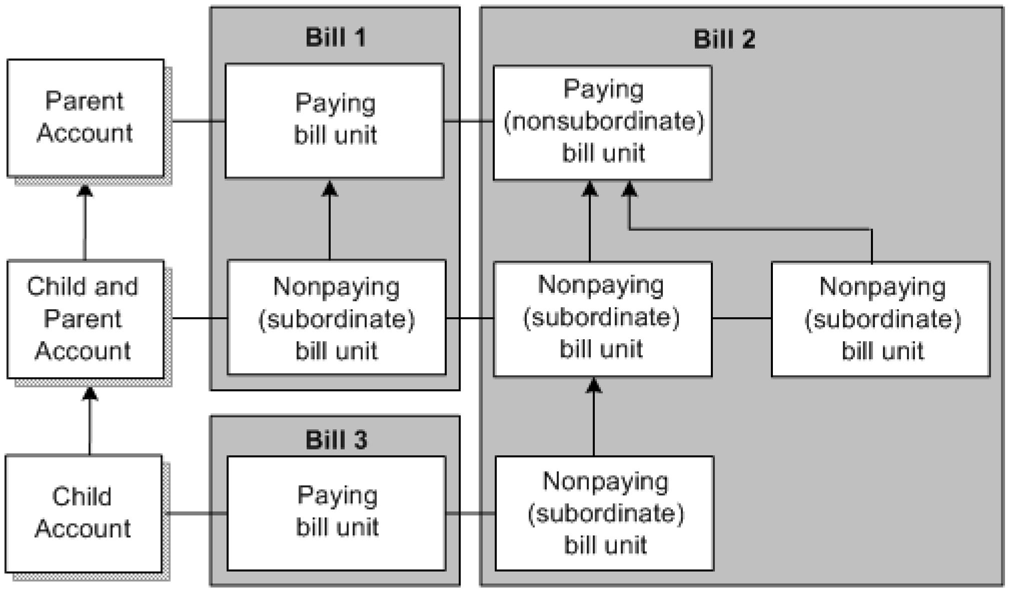
Figure 3–3 Complex Bill Unit Hierarchy


Parent Account
Paying bill unit
Paying (nonsubordinate) bill unit
Child Account
Paying bill unit
Nonpaying (subordinate) bill unit

At each level above the bottom of a hierarchy, the child bill units themselves can be parent bill units.

Figure 3–4 shows a three-level account hierarchy for accounts with multiple bill units. Because there are only three paying bill units, there are only three bills. The top parent account receives two bills and the bottom child account receives one bill:

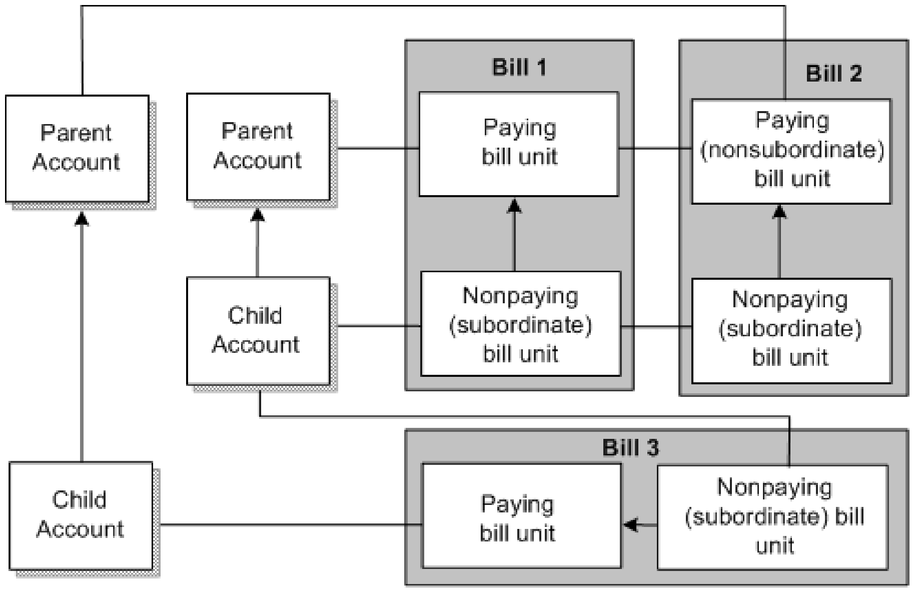
Figure 3–4 Three-Level Account Hierarchy for Accounts with Multiple Bill Units


Bill 1
Bill 2
Parent Account
Paying bill unit
Paying (nonsubordinate) bill unit
Child and Parent Account
Nonpaying (subordinate) bill unit
Nonpaying (subordinate) bill unit
Nonpaying (subordinate) bill unit
Bill 3
Child Account
Paying bill unit
Nonpaying (subordinate) bill unit

A hierarchical bill unit group can contain bill units of accounts that do not belong to the same hierarchical account group.

Figure 3–5 shows two account hierarchies with multiple bill units. Because there are only three paying bill units, there are only three bills. Each parent account receives one bill and only one of the child account receives a bill:

Figure 3–5 Multiple Account Hierarchies with Multiple Bill Units


Bill 1
Bill 2
Parent Account
Parent Account
Paying bill unit
Paying (nonsubordinate) bill unit
Child Account
Nonpaying (subordinate) bill unit
Nonpaying (subordinate) bill unit
Bill 3
Child Account
Paying bill unit
Nonpaying (subordinate) bill unit

To create multiple bill unit hierarchies, see "Creating Hierarchical Bill Units".

How Bill Unit Status Changes Affect Hierarchies

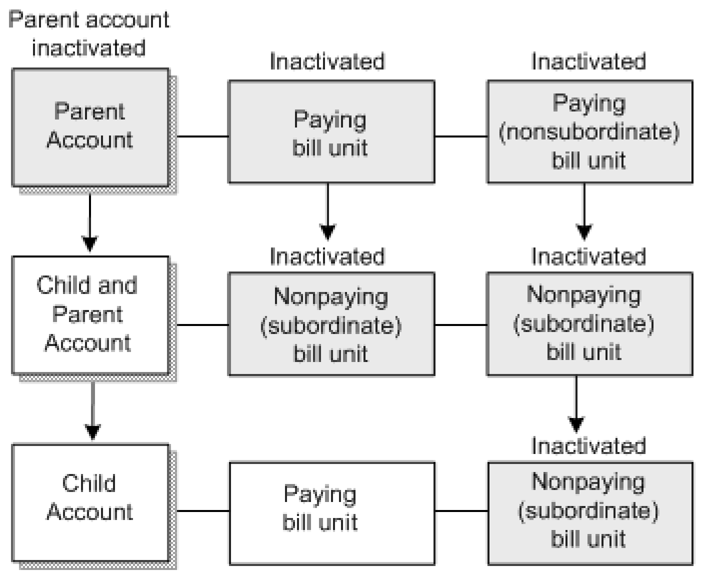
Changing the status of a parent account in a hierarchical group does not affect the status of child accounts; it only affects the status of the subordinate (nonpaying) bill units in the child accounts.

By default, changing the status of the parent account changes the status of all the parent's bill units and those of its child accounts. If a child account is also the parent of another account, the status of all subordinate bill units in each subsequent child account is also changed.

A bill unit's status can be active, inactive, or accounting only. The accounting only status indicates that limited billing is provided, without generating a bill.

For example, Figure 3–6 shows the bill units that are changed to inactive in a multilevel hierarchy when the parent account status is changed to inactive:

Figure 3–6 Impact of Parent Account Inactivation on Bill Units

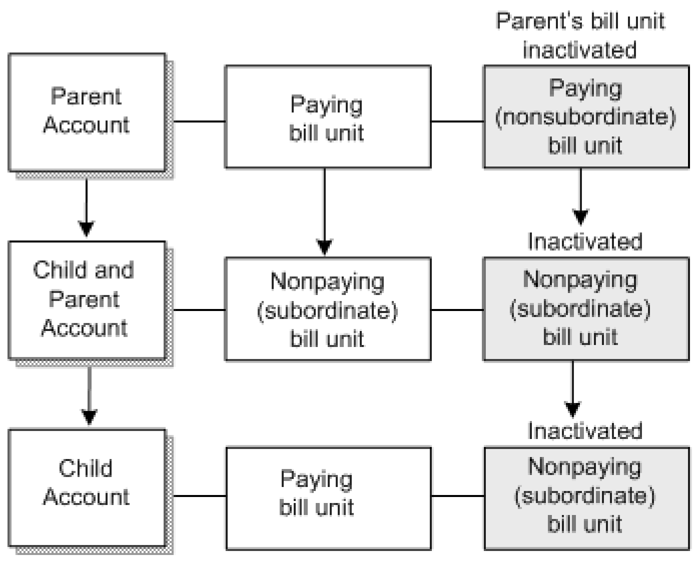

Parent account inactivated
Inactivated
Inactivated
Parent Account
Paying bill unit
Paying (nonsubordinate) bill unit
Inactivated
Inactivated
Child and Parent Account
Nonpaying (subordinate) bill unit
Nonpaying (subordinate) bill unit
Inactivated
Child Account
Paying bill unit
Nonpaying (subordinate) bill unit

When the status of the parent bill unit is changed to accounting only, the status of all its child bill units are also changed to accounting only.

Changing the status of an individual bill unit within a parent account only changes the status of bill units in child accounts that are subordinate to the parent bill unit.

For example, Figure 3–7 shows the bill units that are changed to inactive in a multilevel hierarchy when the status of an individual bill unit in the parent account is inactivated:

Figure 3–7 Impact of Parent Account Inactivating One Bill Unit


Parent's bill unit inactivated
Parent Account
Paying bill unit
Paying (nonsubordinate) bill unit
Inactivated
Child and Parent Account
Nonpaying (subordinate) bill unit
Nonpaying (subordinate) bill unit
Inactivated
Child Account
Paying bill unit
Nonpaying (subordinate) bill unit

Currency Requirements of Hierarchies

Nonpaying bill units in child accounts must have the same currency as their parent account. If the parent and child accounts use two currencies, their primary and secondary currencies must match.

To determine the currency or currencies an account uses, in Customer Center, open the account and look on the toolbar. The first currency in the drop-down list is the primary currency as shown in Figure 3–8:

Figure 3–8 Account Currencies

Billing Setups in Hierarchies

Because paying bill units in parent accounts handle the billing for nonpaying bill units in child accounts, nonpaying bill units must have the same billing day of month, billing frequency, accounting cycle, and language as their parent account.

Paying bill units in child accounts, however, do not need to have the same billing setup and language as their parent account.

A/R and Hierarchical Account Groups

An account group consists of one parent account and one or more child account. When accounts are set up in a hierarchical relationship, each bill unit in the accounts are also assigned the status of parent or child. When accounts are billed, the bill units, not the accounts, are the paying and nonpaying entities.

Child accounts can have a subordinate bill unit (a paying bill unit) or nonsubordinate bill unit (a nonpaying bill unit).

A/R is billed differently for accounts that have nonpaying bill units and paying bill units:

- The A/R for a paying bill unit is billed to itself.
- The A/R for a nonpaying bill unit is billed to the A/R bill unit of the A/R account.

All accounts, both A/R and children with nonpaying bill units, are billed. Billing, however, involves two different processes:

- Changing bills and bill items to open status, and creating new pending bills and bill items for the next bill. This occurs for all accounts.

  Note: If a bill unit in a child account is billed before its parent bill unit, Customer Center displays the nonpaying bill unit's items for the child account as pending until the paying bill unit in the parent account has been billed.

- Requesting a payment for the bill, for example, initiating a credit card transaction. This occurs for paying bill units only. A nonpaying bill unit in a child account never receives a payment request.

BRM creates bills and bill items for all account bill units, including parent, child, nonpaying, and paying bill units. Each bill unit includes a pending bill, and one or more pending bill items. As an account incurs balance impacts, such as usage fees, the amount due accumulates in the pending bill items.

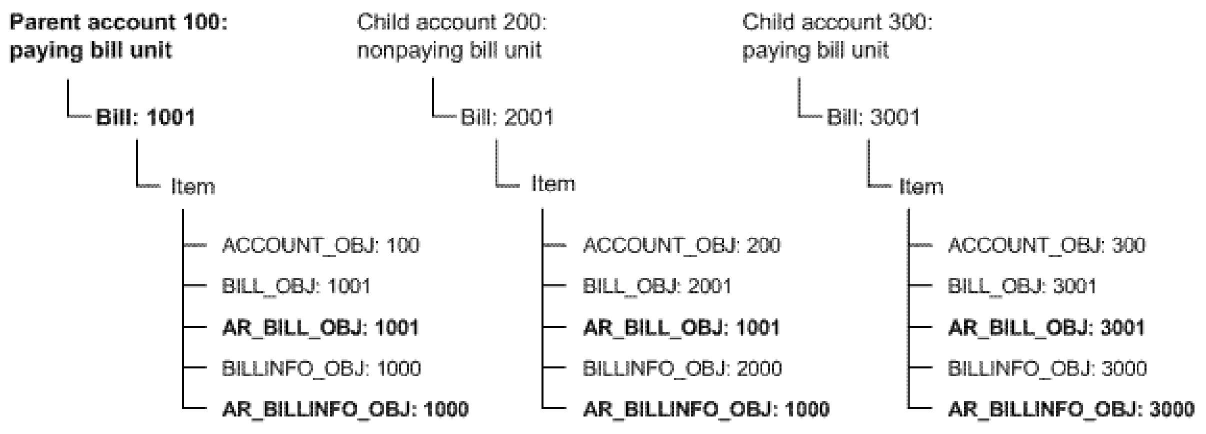
Each bill item includes these fields:

- The account that the item belongs to. This field always points to the account that owns the item.
- The balance group that the item belongs to.
- The bill unit that the item belongs to.
- The bill that the item belongs to. This field always points to the account's own bill.
- The A/R bill unit that the item belongs to.
- The A/R bill that the item belongs to.

The A/R fields determine which account, bill unit, and bill handles billing for the items. If the account has a nonpaying bill unit, the A/R bill unit field points to the parent account's paying bill unit, and the A/R bill field points to a bill in the parent account. In a multilevel hierarchy, the A/R belongs to the top-level, paying bill unit and bill.

Figure 3–9 shows a parent account, a child account with a nonpaying bill unit, and a child account with a paying bill unit. Notice that in the child account with the nonpaying bill unit, the A/R bill field (AR_BILL_OBJ) and A/R bill unit field (AR_BILLINFO_OBJ) point to the parent account.

Figure 3–9 A/R Bill Unit Differences for Child Accounts



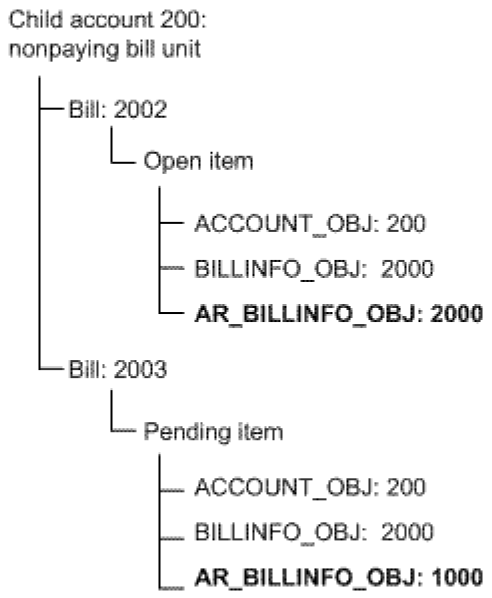
A/R for Open Items and Pending Items

An account might have *open* items and *pending* items. This might happen if the customer hasn't paid an open bill, or has paid only part of it. In that case, there would be an open item and a pending item.

When the account's bill unit changes from a nonpaying to paying, or from paying to nonpaying, the A/R responsibility only for *pending* items is affected. A/R responsibility for *open* items is not changed.

Figure 3–10 shows a child account with a nonpaying bill unit that was once a parent account with a paying bill unit, so it has a pending item and an open item. Notice that the open item points to the child account as the owner of the A/R bill unit (AR_BILLINFO_OBJ) but the pending item points to the parent account as the owner of the A/R bill unit.

Figure 3–10 Account with Pending and Open Items



In some cases, an account can accumulate balance impacts for part of a billing cycle before becoming a child account with a nonpaying bill unit. When the account becomes a child account, the parent account becomes responsible for pending charges accumulated before the child account's bill unit became nonpaying.

Figure 3–11 shows an account that is billed on the 5th day of each month. The account becomes a child account with a nonpaying bill unit on the 10th day of the month, but since it has already incurred balance impacts that are recorded in a pending item, the parent account's bill unit is billed for the A/R for the 5th through the 15th.

Figure 3–11 Child Account Charges before Becoming a Subordinate Account

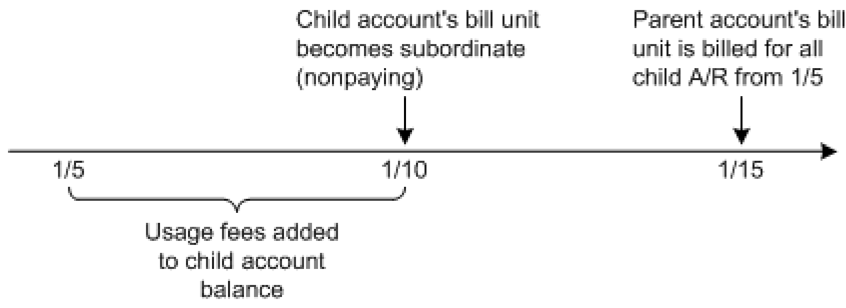

Child account's bill unit becomes subordinate (nonpaying)
Parent account's bill unit is billed for all child A/R from 1/5
1/5
1/10
1/15
Usage fees added to child account balance

Table 3–1 summarizes changes to A/R responsibilities:

Table 3–1 Effect of Changes to Account Hierarchy

Change to Account	Open Items	Pending Items
Parent account with paying bill unit becomes child account with nonpaying bill unit.	Nonpaying bill unit in the child account is responsible for its open item A/R.	Paying bill unit in the parent account is responsible for the pending item A/R of the child account's nonpaying bill unit.
Nonpaying bill unit in a child account becomes a paying bill unit.	Paying bill unit in the parent account is responsible for the open item A/R of the former child account's nonpaying bill unit.	Paying bill unit in the child account is responsible for its pending item A/R.
Child account with nonpaying bill unit changes parent account.	Paying bill unit in the old parent account is responsible for the open item A/R of the child account's nonpaying bill unit.	Paying bill unit in the new parent account is responsible for the pending item A/R of the child account's nonpaying bill unit.

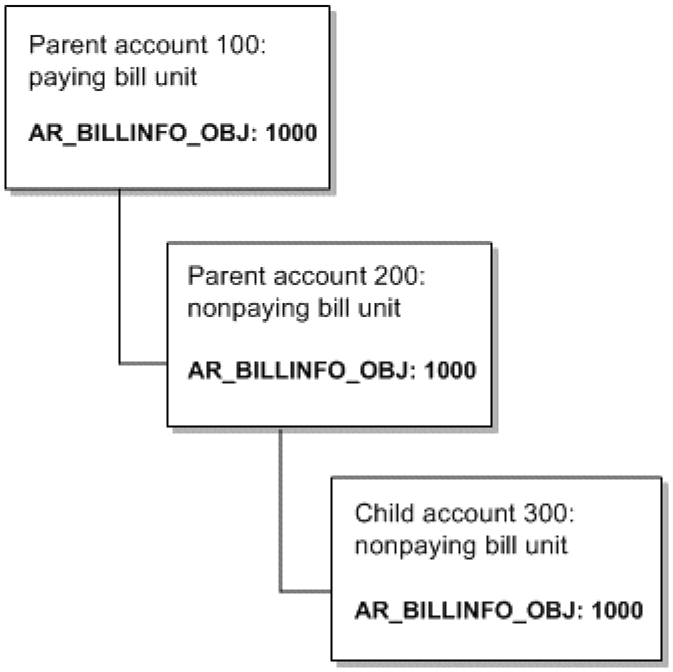
To close all of an account's open items before you make the account a nonpaying child, use the Bill Now feature in Customer Center. For more information, see the Customer Center Help.

Multiple Levels of Parent Accounts

You can have child accounts with nonpaying bill units that are also parent accounts. In this case, the A/R for pending items is always handled by the top parent account in the hierarchy.

In Figure 3–12, account 300, which has a nonpaying bill unit, is a child to account 200, which is in turn a child to account 100. Notice that the A/R for account 300 is handled by account 100, not by account 200.

Figure 3–12 Multiple Levels of Parent Accounts



Hierarchy Changes and Billing Dates

A nonpaying bill unit in a child account must have the same billing day of month as the parent account. When you make an account a child account, the BRM system changes its billing day of month to match the parent account's billing day of month.

When an account becomes a child, its next billing date for the nonpaying bill unit might not be the same as the next billing date of the parent account. This happens because a change to a billing day of month always occurs after the end of the current cycle.

The result can be that the parent account's bill unit is billed, but the nonpaying bill unit in the child account is not billed. Therefore, the A/R for the nonpaying bill unit in the child account is not included in the parent account bill.

For example:

1. You create Account A on June 8.

   The billing day of month is 8.

2. You create Account B on June 20 and then change its billing day to 8.

   The next billing date is August 8. This happens because the current billing cycle, from June 20 through July 20, must be completed before the billing day of month changes. A long billing cycle; from June 20 through August 8, is created.

3. You make Account B a child account to Account A, changing Account B's bill unit to subordinate (nonpaying).

4. On July 8, Account A is billed, but Account B is not because Account B's next billing date is August 8.

   Therefore, the bill only includes the A/R for Account A, the parent account.

5. On August 8, both accounts are billed.

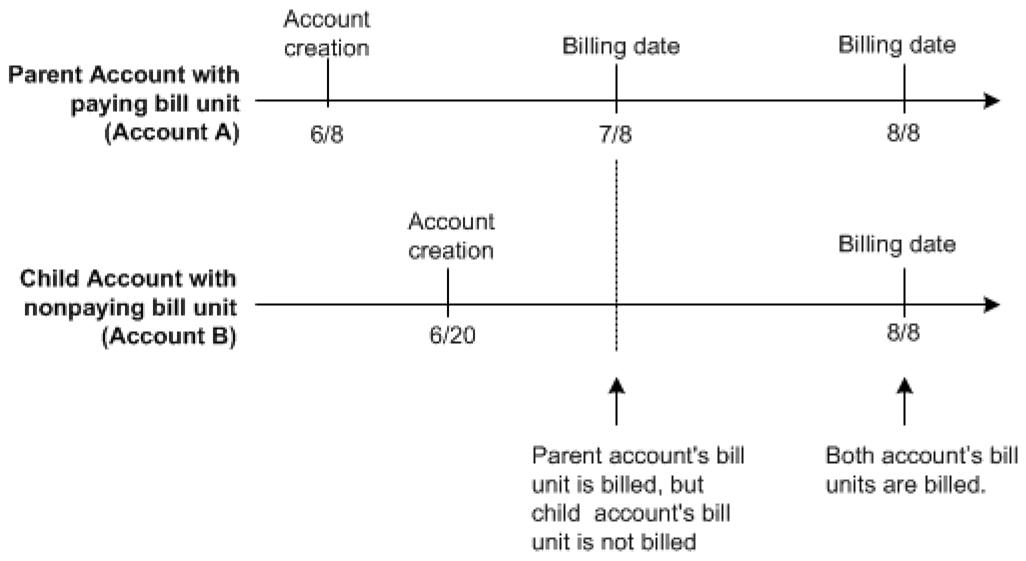
The bill includes A/R from both accounts, including:

- Parent account A/R from July 8 to August 8.
- Child account A/R from June 20 to August 8.

6. On all subsequent billing dates, both accounts are billed.

Figure 3–13 shows a timeline for the billing dates for two accounts in a parent-child relationship where the child account has a nonpaying bill unit.

Figure 3–13 Hierarchy Changes and Billing Dates


Parent Account with paying bill unit (Account A)
Account creation
6/8
Billing date
7/8
Billing date
8/8
Child Account with nonpaying bill unit (Account B)
Account creation
6/20
Billing date
8/8
Parent account's bill unit is billed, but child account's bill unit is not billed
Both account's bill units are billed.

Examples of Changes to Account Group Hierarchy

These examples show who is responsible for A/R for different account hierarchy changes.

An Account Becomes a Child Account with a Paying Bill Unit

In this case, there is no change to how A/R is handled. The parent account handles its own A/R, the child account handles its own A/R.

An Account Becomes a Child Account with a Nonpaying Bill Unit When It Is Created

In this case, all the A/R is handled by the parent account, and the billing date of the child account is the same as the parent account.

An Account Becomes a Child Account with a Nonpaying Bill Unit Immediately After It Is Created

In this case, the account has not been billed yet, so all of its items are pending. Therefore, all of its A/R is handled by the parent account.

Note: By default, for customers who pay by credit card, BRM charges purchase fees and the first cycle forward fee at registration. The A/R for these fees is stored in open items and is charged to the child account's bill unit. You can turn off credit card collection at account creation by editing the **cc_collect** entry in the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**, where *BRM_Home* is the directory in which BRM is installed). See "Charging Customers at Registration" in *BRM Managing Customers*.

An Account Becomes a Child Account with a Nonpaying Bill Unit Several Months After It Is Created

In this case, the A/R for all of the pending items in the nonpaying bill unit of the child account are handled by the parent account. Any open items are handled by the child account. The billing date of the nonpaying bill unit in the child account is changed to match the billing date of the parent account.

A Child Account with a Nonpaying Bill Unit Changes Parent Accounts

In this case, the A/R for all pending items in the nonpaying bill unit of the child account is now handled by a different parent account. Any open items are the responsibility of the former parent account. The billing date of the child account is changed to match the billing date of the new parent account.

A Child Account with a Nonpaying Bill Unit Becomes a Child Account with a Paying Bill Unit

In this case, the fields in pending items that specify the A/R bill unit in the A/R account and the bill both point to the paying bill unit in the child account. If the parent account's bill unit has any open items that include A/R from the child account, the parent account's bill unit is responsible for that A/R even when the child account's bill unit is no longer nonpaying (subordinate).

Creating Hierarchical Groups

By default, accounts are created with one bill unit. When you create account hierarchies in Customer Center, the bill units are automatically assigned the same hierarchical position as the accounts to which they belong. To create account hierarchies, see "Creating Hierarchical Account Groups".

If you want your customers to be able to receive multiple bills for different services, you can create additional bill units per account. When accounts in a hierarchy have multiple bill units, you must also create bill unit hierarchies. See "Creating Hierarchical Bill Units".

Creating Hierarchical Account Groups

There are two ways to create a child account:

- While creating an account in Customer Center, designate an existing account to be its parent. For information about creating accounts, see the Customer Center Help.

Note: In the Customer Center Account Creation wizard, you can create either paying or nonpaying child accounts. The paying and nonpaying designation is assigned to the account's default bill unit when the account is created.

- After creating a standalone account, make it a child account. For more information, see the Customer Center Help.

Creating Hierarchical Bill Units

You create a bill unit hierarchy when you set up account hierarchies in Customer Center. You specify which of an account's bill unit is the paying and non-paying bill unit.

The bill units of accounts that do not belong to the same hierarchical account group can form a bill unit hierarchy. Bill units in a child account can be subordinate to bill units in a different parent account.

For information on using BRM opcodes to create bill unit hierarchies in custom code, see "Managing Bill Units with Your Custom Application" in *BRM Configuring and Running Billing*.

Managing Hierarchical Groups

To manage hierarchies, you can do the following:

- Display hierarchical groups in Customer Center.

  You can see the structure of a hierarchical group in the **Hierarchy** tab of the parent account of the group. Initially, this tab shows the direct lineage of an account, not its siblings or the siblings of its parent.

  For information, see the Customer Center Help.

- Change the parent of an account.

  You can use Customer Center to change an account's parent at any time. You change an account's parent by adding the account to a group, removing it from a group, or moving it from one group to another.

  For information, see the Customer Center Help.

- Defer account group changes until a later date.

  In Customer Center, you can schedule a parent change for a future date. You can then use a daily billing utility, **pin_deferred_act**, to execute the change automatically on the scheduled date.

  For information, see the Customer Center Help and "Managing Deferred Actions" in *BRM Managing Customers*.

Moving Closed Accounts into or out of Hierarchies

By default, BRM does not allow moving closed accounts into or out of account hierarchies. To allow moving of closed accounts into or out of hierarchies, configure the **allow_move_close_acct** entry in the Connection Manager (CM) configuration file.

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **allow_move_close_acct** entry to **1**.

By default, this entry is set to **0**.

```
- fm_bill allow_move_close_acct 1
```

3. Save and close the file.

To remove an account from a hierarchy, see the Customer Center Help.

Managing Hierarchical Account Groups by Using Your Custom Application

To create and manage account groups, use the PCM_OP_BILL_GROUP opcodes, for example, PCM_OP_BILL_GROUP_CREATE (see *BRM Developer's Reference*). These opcodes call the PCM_OP_GROUP opcodes to preform the functionality. Your custom application should use the PCM_OP_BILL_GROUP opcodes instead of calling the PCM_OP_GROUP opcodes directly.

- PCM_OP_BILL_GROUP_CREATE

  See "Creating an Account Group".
- PCM_OP_BILL_GROUP_ADD_MEMBER

  See "Adding a Member to an Account Group".
- PCM_OP_BILL_GROUP_DELETE_MEMBER

  See "Deleting a Member from an Account Group".
- PCM_OP_BILL_GROUP_MOVE_MEMBER

  See "Moving a Group Member".
- PCM_OP_BILL_GROUP_DELETE

  See "Deleting an Account Group".
- PCM_OP_BILL_GROUP_GET_CHILDREN

  See "Getting a List of Child Accounts in an Account Group".
- PCM_OP_BILL_GROUP_GET_PARENT

  See "Finding the Parent of an Account Group".

Creating an Account Group

To create an account group, use PCM_OP_BILL_GROUP_CREATE.

This opcode creates a **/group** object for use in the account group hierarchy, and sets any **/group** object attributes necessary for account billing. When account groups are created for billing purposes, the PIN_FLD_TYPE_STR in the **/group** object is set to **Billing Hierarchy**, and the storable object POIDs created are type **/group/billing**.

The required input fields for PCM_OP_BILL_GROUP_CREATE are:

- The name of the **/group** object
- The parent **/account** object to own the group

To create a **/group** object, PCM_OP_BILL_GROUP_CREATE calls PCM_OP_GROUP_CREATE_GROUP. After the group is created, the new POID is written to the **/account** object of the group's parent (set with the PIN_FLD_PARENT field). PIN_FLD_GROUP_OBJ is the field set in the account record. The output of PCM_OP_GROUP_CREATE_GROUP is returned as the output of this call.

This operation is performed inside a transaction.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_CREATE includes:

- The POID of the group that was created
- A PIN_FLD_RESULTS array with a single event element
- The results array returns the POID of an event storable object entry that holds the old value of the parent storable object and the new value respectively. Since a storable object is being created and an old value did not exist, the old value is always returned as NULL. The old and new parent values are kept in EVENT_ GROUP_PARENTS_T table, which is linked to the EVENT_T table.

If an error occurs, a NULL flist is returned.

Adding a Member to an Account Group

To add a member to an account group, use PCM_OP_BILL_GROUP_ADD_MEMBER.

This opcode 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.

The input to PCM_OP_BILL_GROUP_ADD_MEMBER includes:

- The **/group** object being changed
- A set of account POIDs identifying the accounts to be added

Any account can be added to a given account group, including the parent of another group. PCM_OP_BILL_GROUP_ADD_MEMBER calls PCM_OP_GROUP_ADD_ MEMBER to actually add the new member. After accounts are successfully added, an **/event** object is created that contains a list of the new accounts. The POID of the new **/event** object is returned.

Only new members are actually added to an account group. If the add members list contains accounts that are already in the account group, they are ignored. All accounts being added to a account hierarchy *must*:

- Be unique in that they are not already members of a account group
- Have the same bill time information (NEXT_BILL_T field in the **/billinfo** object) as the parent **/billinfo** object in the parent account for the group

This operation is performed inside a transaction.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_ADD_MEMBER includes:

- The POID of the modified group
- A PIN_FLD_RESULTS array with a single event element

The results array returns the POID of an event storable object entry that holds the list of new accounts added to the account group. The list of new accounts added are kept in EVENT_GROUP_MEMBERS_T table which is linked to the EVENT_T table.

If an error occurs, a NULL flist is returned.

Deleting a Member from an Account Group

To delete a member from an account group, use PCM_OP_BILL_GROUP_DELETE_MEMBER.

This opcode deletes accounts from account groups that are set up for billing purposes.

The input to PCM_OP_BILL_GROUP_DELETE_MEMBER is:

- The group POID
- A set of account POIDs identifying the members to be deleted

The deletion operation is performed using a call to PCM_OP_GROUP_DELETE_MEMBER. After accounts are successfully deleted from the group, an **/event** object is created that contains the list of the accounts deleted. The POID of that **/event** object is returned.

The list of members to delete can contain account POIDs that do not exist in the account group; these members are ignored. Only existing members are deleted from the account group.

A member cannot be deleted from an account group if it has a subordinate bill unit (**/billinfo** object) as subordinate **/billinfo** objects must always be linked to a parent account.

This operation is performed inside a transaction.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_DELETE_MEMBER includes:

- The POID of the group
- A PIN_FLD_RESULTS array with a single event element

The results array returns the POID of an event storable object entry that holds the list of accounts deleted from the account group. The list of accounts deleted are kept in EVENT_GROUP_MEMBERS_T table which is linked to the EVENT_T table.

If an error occurs, a NULL flist is returned.

Moving a Group Member

To move an account from one group to another, use PCM_OP_BILL_GROUP_MOVE_MEMBER.

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

When a member is moved, PCM_OP_BILL_GROUP_MOVE_MEMBER calls PCM_OP_BILL_GROUP_DELETE. If the member being moved is the last member of a group, PCM_OP_BILL_GROUP_MOVE_MEMBER also calls PCM_OP_BILL_GROUP_DELETE to remove the group.

The PIN_FLD_FLAGS field in the PCM_OP_BILL_GROUP_MOVE_MEMBER opcode's input flist controls the type of information returned in the output flist. The settings for this flag are the following:

- **1:** If a bill unit (**/billinfo** object) of the account to be moved is a member of a collections group, the opcode returns the details of the collections group for all the bill units of the account. In this case, the account will not be moved to billing hierarchy.

- **2:** If a bill unit (**/billinfo** object) of an account to be moved is a member of a collections group, the opcode calls the PCM_OP_COLLECTIONS_DELETE_GROUP_MEMBER opcode to delete the collections group member and then continues with moving the bill group member.

If the member is being moved into an existing group, PCM_OP_BILL_GROUP_MOVE_MEMBER also calls PCM_OP_BILL_GROUP_ADD_MEMBER to add it to the target group.

If the member is being moved into a group that does not already exist, PCM_OP_BILL_GROUP_MOVE_MEMBER calls PCM_OP_BILL_GROUP_CREATE to create it.

If the target account (the account heading the nonexistent target group) also does not exist, PCM_OP_BILL_GROUP_MOVE_MEMBER attempts to remove the source member from its group and leave it as a standalone account. If the source member is has a subordinate bill unit (**/billinfo** object), nothing is done and an error is returned.

Note: Any accounts that have bill units that are subordinate to bill units in the moved account are moved with it.

Deleting an Account Group

To delete an account group, use PCM_OP_BILL_GROUP_DELETE.

This opcode deletes account groups that were set up for billing purposes. It performs the following actions:

- Deletes the **/group** object specified.
- Deletes the members list for that account group.
- Clears the PIN_FLD_GROUP_OBJ field in the group parent's **/account** object.

To delete a **/group** object, call PCM_OP_GROUP_DELETE_GROUP. The input to PCM_OP_BILL_GROUP_DELETE is the POID of the account group to be deleted. After successfully deleting the group, it returns the old group POID.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_DELETE includes the POID of the group that was deleted.

If an error occurs, a NULL flist is returned.

Getting a List of Child Accounts in an Account Group

To get a list of child accounts in an account group, use PCM_OP_BILL_GROUP_GET_CHILDREN.

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.

This operation is performed inside a transaction.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_GET_CHILDREN includes:

- The POID of the group that was modified.
- An array of PIN_FLD_MEMBERS, where each array element holds an account POID along with one or more account fields for that entry.

If an error occurs, a NULL flist is returned.

Finding the Parent of an Account Group

To find the parent of an account group, use PCM_OP_BILL_GROUP_GET_PARENT.

PCM_OP_BILL_GROUP_GET_PARENT retrieves the parent account of a given **/group** object. The input to PCM_OP_BILL_GROUP_GET_PARENT is a account group POID. The account POID identifying the group’s parent account is returned.

This operation is performed inside a transaction.

Additional Return Information

The output flist for PCM_OP_BILL_GROUP_GET_PARENT includes the POID of the account that is the parent of the group.

If an error occurs, a NULL flist is returned.

About the PCM_OP_GROUP Opcodes

The Group FM includes the following standard opcodes. These opcodes are called by other opcodes, for example, the wrapper opcode PCM_OP_BILL_GROUP_CREATE calls the PCM_OP_GROUP_CREATE_GROUP opcode to create a group. Whenever possible, use the wrapper opcode, not the GROUP opcode.

- PCM_OP_GROUP_CREATE_GROUP

  See "Creating a Group".
- PCM_OP_GROUP_ADD_MEMBER

  See "Adding Members to a Group".
- PCM_OP_GROUP_DELETE_MEMBER

  See "Deleting Members from a Group".
- PCM_OP_GROUP_SET_PARENT

  See "Setting a Group Parent".
- PCM_OP_GROUP_DELETE_GROUP

  See "Deleting a Group".
- PCM_OP_GROUP_UPDATE_INHERITED

  See "Updating the Inheritance Fields in a Group".

Creating a Group

To create a new group storable object, use PCM_OP_GROUP_CREATE_GROUP.

After successfully creating the group, the new group POID is returned.

This operation is performed inside a transaction.

PCM_OP_GROUP_CREATE_GROUP first initializes the group to be created. Then this opcode uses PCM_OP_GROUP_SET_PARENT to set any parent information specified on input. If a members list is also passed in, PCM_OP_GROUP_ADD_MEMBER is called to add those members. PCM_OP_GROUP_CREATE_GROUP does not generate an event storable object. However, if new members are specified and added to the group, they will create an **/event** object. The output of PCM_OP_GROUP_SET_PARENT and PCM_OP_GROUP_ADD_MEMBER are attached to the return flist.

Adding Members to a Group

To add members to a group, use PCM_OP_GROUP_ADD_MEMBER.

The input to this opcode is a set of member POIDs identifying the members to be added. After members are successfully added, an **/event** object is created containing a list of new members added. The POID of the **/event** object is returned.

This operation is performed inside a transaction.

The add members list may contain member POIDs that already exist in the group. These members are ignored; only new members are added to the group.

Deleting Members from a Group

To delete members from a group, use PCM_OP_GROUP_DELETE_MEMBER.

The input to this opcode is a set of member POIDs identifying the members to be deleted. After successfully deleting members, an **/event** object is created containing a list of the members deleted. The POID of the **/event** object is returned. This operation is performed inside a transaction.

The delete-members list may contain member POIDs that do not exist in the group. These elements are ignored and not processed; only existing members are deleted from the group.

Setting a Group Parent

To set the parent storable object of a group, use PCM_OP_GROUP_SET_PARENT.

This opcode sets the parent storable object for the given group. The input to PCM_OP_GROUP_SET_PARENT includes the group POID to be modified and the new parent storable object to be set. After successfully setting the parent field, the group POID id is returned.

This operation is performed inside a transaction.

The parent field is specified using a POID id, which is similar to a member element of a group. However, a parent is considered the storable object that has ownership of this group. If this group represents accounts, then the members elements refer to accounts that belong to this group and the parent storable object would then be an account that is responsible for all the member accounts in the group.

The results array returns the POID of an **/event** object entry that holds the old value of the parent storable object, and the new value, respectively. The old and new parent values are kept in the EVENT_GROUP_PARENTS_T table, which is linked to the EVENT_T table.

Deleting a Group

To delete an existing group storable object, use PCM_OP_GROUP_DELETE_GROUP.

PCM_OP_GROUP_DELETE_GROUP removes the specified **/group** object and all members linked to it. The input to this opcode is the POID of the group to be deleted. After successfully deleting the group, the old group POID id is returned.

This operation is performed inside a transaction.

Updating the Inheritance Fields in a Group

To update the inheritance fields of an existing group, use PCM_OP_GROUP_ UPDATE_INHERITED.

The input to this opcode is the group POID to be updated, followed by the inheritance information. The inheritance information is passed as a substructure flist and must have space allocated for it as a separate storable object, independent of the **/group** storable object.

This operation is performed inside a transaction.

4

Managing Resource Sharing Groups

This chapter describes how to create and manage charge sharing and discount sharing groups in your Oracle Communications Billing and Revenue Management (BRM) system.

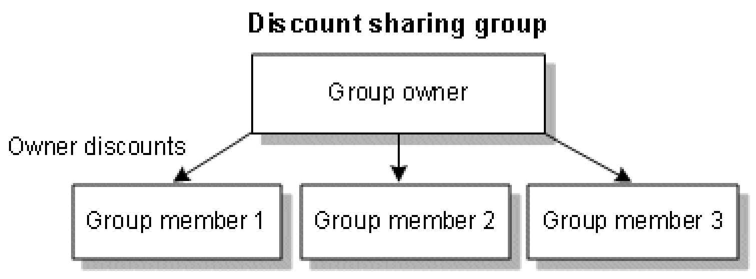
For information about different types of groups, see "About Account Groups".

About Resource Sharing Groups

A resource sharing group consists of an owner and one or more members who share resources. A resource sharing group can be one of the following types:

- **Discount sharing group**. In a discount sharing group, the owner shares its discounts or resources with the members, as shown in Figure 4–1:

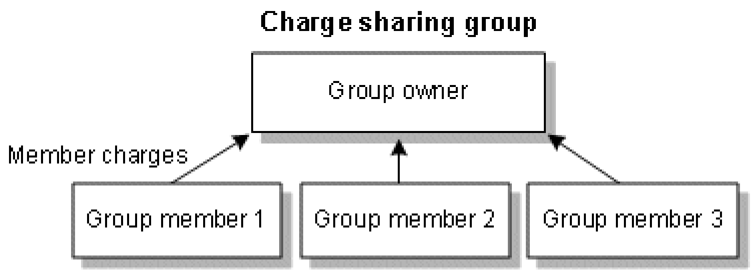
Figure 4–1 Discount Sharing Group



For more information, see "About Discount Sharing Groups".

- **Charge sharing group**. In a charge sharing group, the owner assumes charges that are incurred by the members, as shown in Figure 4–2:

Figure 4–2 Charge Sharing Group



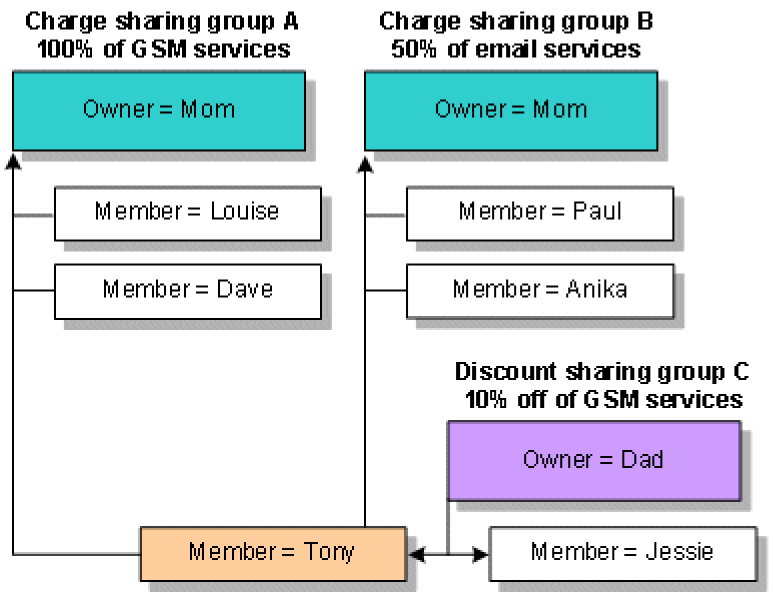
For more information, see "About Charge Sharing Groups".

You can use Customer Center to set up discount and charge sharing groups, or you can customize a third-party client application to set up resource sharing. For information on using Customer Center to create and manage resource sharing groups, see the Customer Center Help. For information on customizing resource sharing, see "Managing Resource Sharing Groups".

Working with Complex Resource Sharing Groups

Networks of resource sharing groups can be complex. An owner can have more than one resource sharing group, and a member can belong to more than one resource sharing group, as shown in Figure 4–3:

Figure 4–3 Complex Resource Sharing Groups



In this example, Mom owns two charge sharing groups: one for GSM services and one for email services. She assumes charges for all her children; 100% of GSM charges for Louise and Dave, 50% of email charges for Paul and Anika, and both GSM and email charges for her youngest son, Tony. Dad owns a discount sharing group: he shares a 10% discount on GSM charges with his son Tony and brother Jessie. The members have the following benefits:

- When Paul and Anika use email, Mom assumes part of their charges. They have no discounts, shared or owned, so Mom pays 50% of their monthly email charges and they pay the remaining 50%.
- When Louise and Dave use GSM services, Mom assumes their charges. They have no discounts, so Mom pays 100% of their monthly GSM charges.
- When Jessie uses GSM services, he receives a 10% discount from Dad. Jessie is not a member of any charge sharing groups, so he pays the remaining 90% of his GSM charges.
- When Tony uses email, Mom assumes part of his charges. He does not have any email discounts, so Mom pays 50% of his monthly email charges and he pays the remaining 50%.

- When Tony uses GSM services, he receives a 10% discount from Dad. This discount is applied before the charges are applied. Because Tony is a member of charge sharing group A, Mom pays the remaining 90% of his GSM charges.

In more complicated networks, an owner of a resource sharing group can also be a member of another resource sharing group. For example, Mom owns a charge sharing group that assumes email charges for Paul, Anika, and Tony. Mom, in turn, is a member of Grandma's charge sharing group, which also assumes email charges for its members. In this case, Grandma pays the email charges for Mom. In addition, she inherits the email charges for Paul, Anika, and Tony as a result of Mom's charge sharing group.

Important: When setting up resource sharing groups, you should avoid circular relationships. See "Creating Resource Sharing Groups".

About Discount Sharing Groups

Important: Subscriber Groups (discount sharing groups) are dependent upon the Advanced Discounting Manager (ADM), an optional feature that requires a separate license.

Discount sharing occurs when an account or service shares its discounts with other accounts' services. The account that shares its discounts is the owner of a discount sharing group. Within the owner's account, one of the balance groups serves as the owning balance group. The owning balance group is determined based on whether the discount sharing group owner is the account or a service in the account:

- If the group owner is the account, the owning balance group is the account's default balance group, and all discounts purchased by the account are shared.
- If the group owner is a service in the account, the owning balance group is the associated service-level balance group, and only the discounts for that service are shared.

When a discount sharing group is created, a discount sharing group object (**/group/sharing/discounts**) is added to the database. This object contains the owner account or service, the list of discounts that are shared by the owner, and the list of members. The members are the services that use the shared discounts.

When the discount sharing group owner is a service, its service type must match the type of service to which the shared discounts apply.

When events are generated as a result of member activity, discounts belonging to the sharing group owner are typically applied first, followed by any discounts belonging to the member account. After discounting, any remaining balance impact is applied to the member account unless the account also participates in charge sharing.

Note: You can change the order in which discounts are consumed so that discounts belonging to the member are used before shared discounts.

For members to receive discounts from the owner, the member's ordered balance group must include a reference to the discount sharing group. This reference is created when the member joins the group. For more information, see "How Discounts and

Charges Are Applied".

To set up a discount sharing, see "Setting Up Discount Sharing" in *BRM Configuring Pipeline Rating and Discounting*.

To create a discount sharing group, see "Creating Resource Sharing Groups".

How Account Status Changes Affect Discount Sharing Groups

Changing the status of a discount sharing group's owner account changes the status of all its shared discounts. When the owner account status is changed to inactive or closed, members stop benefiting from the discount sharing group. Members cannot receive any shared currency or percentage discounts. Also, any new events generated by the members impact resources of the member's balance groups *only*; they can no longer take shared resources from the owner's account.

The way that BRM treats the discount sharing group depends on whether the owner account is deactivated or closed:

- **Owner account is deactivated**: The status of all the group's shared discounts are changed to inactive. Member services can begin consuming resources from the group's shared pool again as soon as the owner's account is reactivated.
- **Owner account is closed**: All of the group's shared discounts are removed from the account, and the shared discounts are removed from the discount sharing group.

Note: If **keep_cancelled_products_or_discounts** is set to **1** in the Connection Manager (CM) **pin.conf** file, the shared discounts are not removed from the discount sharing group. Instead, their status is set to canceled. For information about deleting canceled discounts, see "Specifying to Delete Canceled Discounts" in *BRM Managing Customers*.

How Group Owner Changes Affect Discount Sharing Groups

If you use a third-party client application, you can customize the application to enable a customer service representative (CSR) to change the owner of a discount sharing group. For information on how to implement this, see "Changing the Owner of a Discount Sharing Group".

Note: Customer Center does not support changing group owners.

When the owner of a discount sharing group changes, the discounts shared by the previous owner are deleted from the group, and discounts shared by the new owner are added.

Member-generated events, including delayed events, that occur after a group owner is changed impact the new owner's resource balances.

Members and Discount Sharing Groups

When you set up a discount sharing group, the members of the group must be services (the service-level balance groups). Account-level balance groups can only be members in balance monitoring groups (see "About Balance Monitoring").

If the owner of a shared discount is a service, the discount sharing group member service is typically the same service type as, or a subclass of, the discount owner service. For example, if the discount sharing group owner is **/service/telco**, the discount sharing group member might be **/service/telco/gsm**. This parent and subclass relationship is the standard way you set up subscription services to offer a discount based on the total usage of all subscription services in an account group.

If the owner of a shared discount is a service, Customer Center requires that the owner and member services be of the same type. You can specify different service types for the owner and members by implementing the relevant subscription opcodes in your custom code (see "Managing Resource Sharing Groups"). However, if the owner and member services are different types, the member service must match one of the shared discount's permitted service types (specified in the PIN_FLD_PERMITTEDS array in the **/discount** object).

If the owner and a member of the discount sharing group have different service types, to apply the discount to the member service, the member's usage event type must match the event type to which the shared discount applies.

For example, you use Pricing Center to set up a subscription discount that applies to **/service/telco** usage. In the Discount Attributes dialog box, you map the discount model to the monthly cycle forward event. You then create a discount sharing group in which you share the telco discount that you created. You add an account to the discount sharing group and specify the account's **/service/telephony** service as the member. In this case, the subscription discount is applied to the member account only when the member account generates a monthly cycle forward event for its telephony service.

When you set up a discount sharing group, you can specify that the member service be a service type or a service instance:

- **Specify a service type as the member**: When you specify a service type (for example, GSM) as a member, the events generated by all service instances within that service type are considered for discount sharing. Adding members by service type is appropriate when:
  - You are adding multiple member accounts to the group and you want them all to take advantage of discounts for a service type (for example, an aunt who wants to share her GSM discounts with all her nieces and nephews). In this case, each niece and nephew has a different service instance, so instead of specifying each service instance individually, you specify the member accounts and the service type. This captures all of the nieces' and nephews' service instances under the umbrella of the GSM service type.
  - You want a member account that has not yet purchased a service of that type to be automatically eligible for participation in discount sharing if the member buys the service in the future. With future sharing, even though a member becomes automatically eligible, you must track the member's purchases and, when the service is purchased, manually intervene to join discount sharing for the new service instance.

    For information about how future services work in Customer Center, see the Customer Center Help.
- **Specify a service instance as the member**: When you specify a service instance (for example, an employee's work phone) as a member, only the events generated by that instance are eligible for discounting. Specify membership using a specific service instance to exclude other services of that type from discount sharing.

For example, John’s account includes work email (smith@CompanyA.com) and home email (john@internetprovider.com). John’s employer wants to share a 10% discount for monthly fees incurred only by work email. Because John’s account has both home and work services, you designate the service instance for smith@CompanyA.com as the member to prevent discount usage by john@internetprovider.com.

Currency Requirements of Discount Sharing Groups

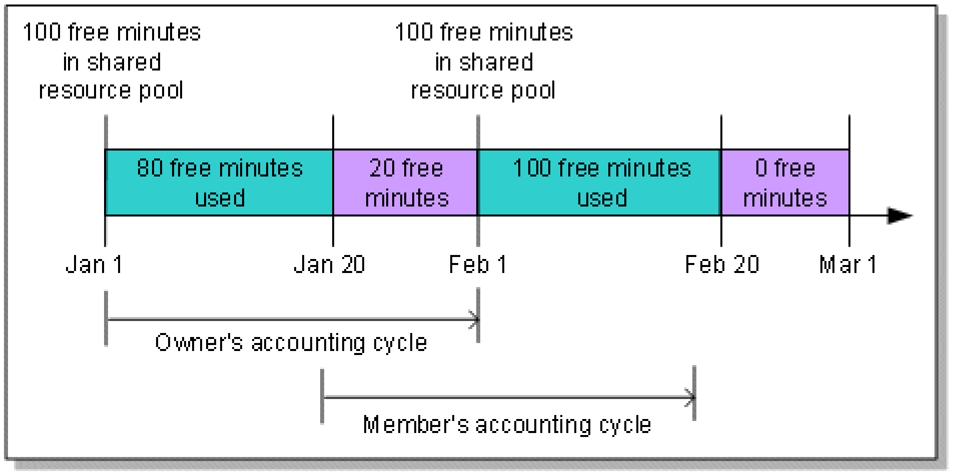
The discount sharing group’s owner account and member accounts must use the same currency. If the accounts use two currencies, they must use the same primary currency.

Billing for Discount Sharing Groups

A discount sharing group’s owner account and member accounts can have different accounting and billing cycles. Member-generated events that qualify for discounts always consume the available resources from the discount sharing group’s shared resource pool.

In the example illustrated in Figure 4–4, the owner’s accounting and billing cycles begin on the 1st of the month, and the member’s accounting and billing cycles begin on the 15th of the month. The owner receives 100 free monthly minutes that are shared.

Figure 4–4 Discount Sharing between Accounts with Different Accounting Cycles



When a member generates an event that lasts 30 minutes on January 20, it consumes the available 20 free minutes from the shared resource pool. The member’s account receives a balance impact for the remaining 10 minutes of the event.

When another member generates an event that lasts 15 minutes on February 20, there are no available free resources in the shared pool to consume, and the member account receives a balance impact for the entire 15 minutes.

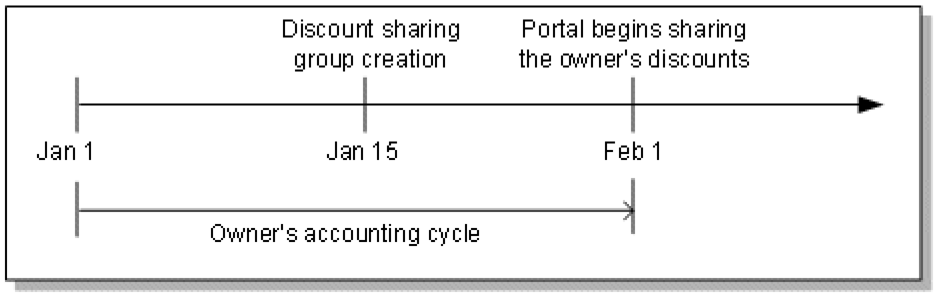
Configuring the Start and End Times for Discount Sharing

By default, discount sharing start and end times are based on the date of the next billing cycle. Therefore:

- When you create a discount sharing group, add a discount to an existing group, or add a member to the group, discount sharing starts at the beginning of the owner's next billing cycle.
- When you delete a discount sharing group, delete a discount from a group, or remove a member from the group, discount sharing is effective up to the end of the owner's billing cycle.

For example, if you create a discount sharing group on January 15 but the next billing cycle does not start until February 1, the members of the group do not receive any of the owner's shared discounts until February 1, as shown in Figure 4–5:

Figure 4–5 Effective Date of Discount Sharing Group Created Mid-Cycle


Discount sharing group creation
Portal begins sharing the owner's discounts
Jan 1
Jan 15
Feb 1
Owner's accounting cycle

Conversely, if you delete a discount sharing group on January 15 but the next billing cycle does not start until February 1, the members of the group receive shared discounts based on the entire month of January, not just the first 15 days.

You can configure BRM to start and end discount sharing as soon as the group is created or deleted, a discount is added or deleted, or a member is added or removed. To do so, you modify an entry in the Connection Manager (CM) **pin.conf** file.

When you configure discount sharing this way, BRM prorates the shared discounts according to where the start or end date falls within the billing cycle. For example, if you delete a discount sharing group on April 15 and the owner's billing cycle starts on the first day of each month, the members receive 50% of the discount, equivalent to the portion of the month in which discount sharing existed.

To change the discount sharing start time:

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf)**.
2. Change the discount sharing startup from the beginning of the next billing cycle to the time when the discount sharing group is created or a member is added by changing the **0** to **1** in the following line:

   ```
   -fm_subscription propagate_discount 1
   ```

3. Stop and restart the CM and, if necessary, your client application.

   For information on restarting the CM, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Creating a Discount Sharing Group for Accounts That Use a Custom Payment Method

If accounts use a custom payment method, you must edit the Customer Center **CustomizedResources.properties** file if you want accounts to own a discount sharing group.

Add the following entry to the Customer Center **CustomizedResources.properties** file:

```
object.payinfo.payinfo_subclass_name.mapping=payment_type_name
```

For example, if the payment type is named Custom Direct Debit and the payment information subclass is **/payinfo/ddebit**, the properties file entry is:

```
object.payinfo.ddebit.mapping=Custom Direct Debit
```

About Charge Sharing Groups

Important: Subscriber Groups (charge sharing groups) is an optional feature that requires a separate license.

Charge sharing enables a customer to sponsor the charges of other accounts or services in the system. For example, it enables a company to pay for all of its employees' GSM telephony services, or a parent to pay for his child's SMS and email services.

You set up charge sharing by creating a charge sharing group, which consists of the following:

- **Charge sharing group owner**: The account or service responsible for all or a portion of the charges incurred by the charge sharing group members.
- **Charge sharing group members**: The accounts and services that the owner sponsors.
- **Chargeshare**: The chargeshare specifies how and when to apply charges. It includes a usage map that links particular event types to chargeshare models. Chargeshare models determine whether events qualify for charge sharing and apply rules to calculate charge sharing amounts.

When a member incurs charges sponsored by the owner, the charges are applied to the owner's balance group first. Any charges that remain afterward impact the member's balance group.

Information about charge sharing groups is stored in **/group/sharing/charges** objects in the BRM database.

To set up charge sharing, see "Setting Up Charge Sharing" in *BRM Configuring Pipeline Rating and Discounting*.

To create a charge sharing group, see "Creating Resource Sharing Groups".

About Charge Sharing Group Owners

The account that sponsors the charges is the *owner* of a charge sharing group. Within the owner's account, one of the balance groups serves as the owning balance group. The owning balance group is the one that receives the balance impact of any shared charges incurred by the members.

The owning balance group is determined based on whether the charge sharing group owner is the account or a service in the account:

- If the group owner is the account, the owning balance group is the account's default balance group.
- If the group owner is a service in the account, the owning balance group is the associated service-level balance group.

About Serviceless Accounts as Charge Sharing Owners

Charge sharing group owners can be accounts that have no services. This enables you to bypass purchasing specific services for corporate accounts that are used exclusively to pool employee charges. For example, you can create a corporate account that assumes telephone, email, and IP charges for all employees in the marketing group. But, you do not have to purchase GSM, email, or IP services for that corporate account. If you set up a serviceless account as a charge sharing group owner, BRM applies the member charges to that account's default balance group.

Pipeline Manager uses the create, modify, and delete sharing group events to determine the owner of the sharing group. If the owner is an account without services, the PIN_FLD_SERVICE_OBJ value is NULL. If the owner is a service, the PIN_FLD_SERVICE_OBJ value is the POID of the service.

To create a charge sharing owner that is a serviceless account, perform the following:

1. Enable Pipeline Manager to accept serviceless accounts by setting the DAT_AccountBatch **LoadAccountForSharingOnly** registry entry to **True**. This prevents the pipeline from rejecting serviceless accounts if they are owners of resource sharing groups. By default, this entry is set to **False.**

   For more information about DAT_AccountBatch, see "DAT_AccountBatch" in *BRM Configuring Pipeline Rating and Discounting*.

2. Create a serviceless account. First use Pricing Center to create a plan that has no deals. Then, purchase the plan when you create the account in Customer Center. When you create a charge sharing group for this account, the result is an account-level charge sharing group not associated with any specific services.

How Owner Account Status Changes Affect Charge Sharing

By default, changing the status of a charge sharing group's owner account changes the status of all its shared charges. When the owner account is deactivated or closed, members stop benefiting from the charge sharing group. The owner no longer assumes any of the charges incurred by the members. Any new events generated by the members are charged to the member's balance groups *only*.

The way that BRM treats the charge sharing group depends on whether the owner account is deactivated or closed:

- **Owner account is deactivated**: All of the group's sponsored charges are suspended, and no new member charges are added to the owning balance group. Member services can begin benefiting from charge sharing again as soon as the owner's account is reactivated.
- **Owner account is closed**: All of the group's shared charges are suspended, and no new member charges are added to the owning balance group. The shared charges are removed from the charge sharing group.

Note: If **keep_cancelled_products_or_discounts** is set to **1** in the Connection Manager (CM) **pin.conf** file, the shared charges are not removed from the charge sharing group. Instead, their status is set to canceled. For information about deleting canceled discounts, see "Specifying to Delete Canceled Discounts" in *BRM Managing Customers*.

About Changing Charge Sharing Group Owners

If you use a third-party client application, you can customize the application to enable a CSR to change the owner of a charge sharing group. For information on how to implement this, see "Changing the Owner of a Charge Sharing Group".

Note: Customer Center does not support changing group owners.

When the owner of a charge sharing group changes, the charges that were sponsored by the previous owner are deleted from the group, and charges sponsored by the new owner are added.

Member-generated events, including delayed events, that occur after a group owner is changed impact the new owner's resource balances.

About Charge Sharing Group Members

When you set up a charge sharing group, the members of the group must be services (the service-level balance groups). Account-level balance groups can be members only in balance monitoring groups (see "About Balance Monitoring").

If the owner of a charge sharing group is a service, the group member service is typically the same service type as, or a subclass of, the charge share owner service. If the owner and a member of the charge sharing group have different service types, to share charges, the member's usage event type must match the event type to which the charge share applies.

Note: Customer Center requires that the owner and member services be of the same type. You can specify different service types for the owner and members by implementing the relevant subscription opcodes in your custom code. See "Managing Resource Sharing Groups".

When you set up a charge sharing group and specify the member services, you can:

- **Specify a service type for a group of accounts as members**: When you specify multiple account instances and a service type (for example, Account 1, Account 2, and GSM) as members, only the service instances for each account are considered members. Adding members by service type is appropriate when:
  - You are adding multiple member accounts to the group and you want them all to take advantage of charge sharing for a service type (for example, an aunt who wants to pay the GSM charges for all of her nieces and nephews). In this case, each niece and nephew has a different service instance, so instead of specifying each service instance individually, you specify the member accounts and the service type. This captures all of the nieces' and nephews' service instances under the umbrella of the GSM service type.
  - You want a member account that has not yet purchased a service of that type to be automatically eligible for participation in charge sharing if the member buys the service in the future. With future sharing, even though a member becomes automatically eligible, you must track the member's purchases and, when the service is purchased, manually intervene to join charge sharing for the new service instance.

For information about how future services work in Customer Center, see the Customer Center Help.

- **Specify a service instance as the member**: When you specify a service instance (for example, an employee's work phone) as a member, only the events generated by that instance are eligible for charge sharing. Specify membership by using a specific service instance to exclude other services of that type from charge sharing.

  For example, John's account includes work email (smith@CompanyA.com) and home email (john@internetprovider.com). John's employer wants to pay for monthly fees incurred by work email only. Because John's account has both home and work services, you designate the service instance for smith@CompanyA.com as the member to prevent the employer from paying for usage charges from john@internetprovider.com.

- **Specify all accounts in the system as members**: When you specify a type-only account POID as the member, events generated by any account in your system are eligible for charge sharing.

  See "About Global Charge Sharing Groups".

- **Specify all services of a specific type as members**: When you specify a type-only service POID as the member, events generated for that service type are eligible for charge sharing.

  See "About Global Charge Sharing Groups".

For member charges to be applied to the owner's account, the member's ordered balance group must include a reference to the charge sharing group. This reference is created when the member joins the group. For more information, see "How Discounts and Charges Are Applied".

Note: This requirement does not apply to global charge sharing groups. See "About Global Charge Sharing Groups".

Currency Requirements for a Charge Sharing Group

The charge sharing group's owner account and member accounts must use the same currency or primary currency.

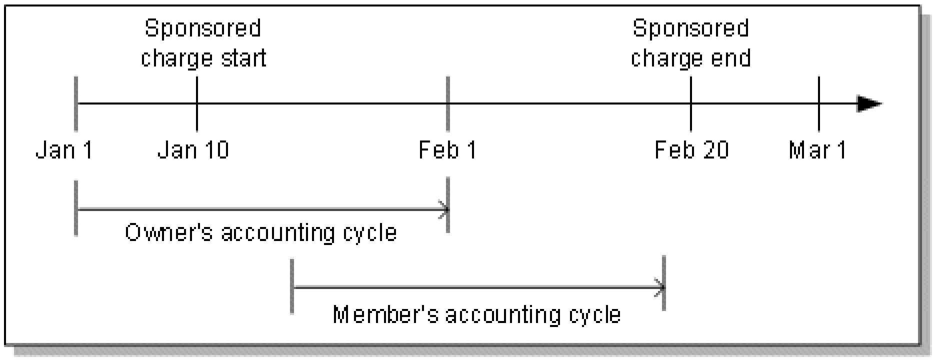
Note: The currency requirement does not apply for global charge sharing groups. See "About Global Charge Sharing Groups".

Billing for a Charge Sharing Group

A charge sharing group's owner and member accounts are not required to have the same accounting or billing cycles. Sponsored charges are always applied to the owner account, regardless of when the member-generated events occurred.

In the example illustrated in Figure 4–6, the owner's accounting and billing cycles begin on the 1st of the month, and the member's accounting and billing cycles begin on the 15th. Charge sharing starts on January 10 and ends on February 20, so between those dates, the owner sponsors 50% of all charges incurred by a member.

Figure 4–6 Effective Dates of Charge Sharing Groups Created Mid-Cycle



The member generates an event on January 20 for 30 minutes at $0.10 per minute, for a total cost of $3.00. The owner's account receives a balance impact of $1.50 and the amount appears on the owner's bill on February 1. The member's account receives a balance impact of $1.50, and the amount appears on the member's bill on February 15.

If the member generates an event on February 10 for 10 minutes, the owner is billed $0.50 on March 1 and the member is billed $0.50 on February 15.

The member receives the balance impacts for all charges incurred after February 20.

About Global Charge Sharing Groups

You can create a charge sharing group that includes all accounts in your system or all services of a specific type, such as GSM telephony. This type of charge sharing group is called a global charge sharing group. This enables one account to pay for all or a portion of the charges incurred by anyone in your system.

You specify which events the group owner pays for and how to split charges between the owner and members by creating a chargeshare. For information, see the Pricing Center Help.

For example, you can configure BRM to charge a company for all or a portion of any call to a toll free number or any text message sent to a specified number. For an example of how to set up global charge sharing groups for toll free numbers, see "Global Charge Sharing Configuration Example" in *BRM Configuring Pipeline Rating and Discounting*.

Global charge sharing groups differ from charge sharing groups in these ways:

- Global charge sharing groups are not listed in a member's ordered balance group. Therefore, the order in which charges are applied to global charge sharing groups is set by the system.
- Members in the global charge sharing group are indicated by a type-only POID.

About the Order in Which Charges Are Applied to Groups

The order in which BRM applies charges to the different types of resource sharing groups is built into the system. BRM automatically applies charges to global charge sharing groups after applying any discounts from discount sharing groups but before applying charges to any charge sharing groups. The order in which BRM applies discounts and charges is shown below:

1. Discount sharing groups as listed in the member's ordered balance group
2. Global charge sharing groups having type-only services

3. Global charge sharing groups having type-only accounts
4. Charge sharing groups as listed in the member's ordered balance group

Note: Only charge sharing groups and discount sharing groups are listed in a member's ordered balance group. Global charge sharing groups are not included in the list.

About Creating Global Charge Sharing Groups

Global charge sharing groups, like charge sharing groups, consist of an owner, members, and the chargeshare that defines when and how to apply charges. However, instead of listing every group member, the global charge sharing group indicates all accounts in the system or all services of a specific type as members by using a type-only POID.

Important: Charge sharing group members must be either a type-only POID or a list of member POIDs; members cannot include both types.

Any existing account or service in your system is automatically considered a part of the group. Likewise, when you create a new account or service in the system, BRM automatically considers the account or service as part of the global charge sharing group.

To create global charge sharing groups, perform the following:

1. Make sure the global charge sharing search is enabled for real-time rating.

   See "Enabling Global Charge Sharing Searches during Discounting".

2. Use Customer Center or a custom client application to create global charge sharing groups:

   - To create global charge sharing groups by using Customer Center, see the Customer Center Help.
   - To create global charge sharing groups by using a custom client application, see "Using Third-Party Client Applications to Create, Modify, and Delete Global Charge Sharing Groups".

Enabling Global Charge Sharing Searches during Discounting

By default, the global charge sharing search is disabled. You can enable global charge sharing searches by changing the **EnableGlobalChargeSharing** field in the **rating** instance of the **/config/business_params** object from **disabled** to **enabled**. You can disable the feature by changing the field back to **disabled**.

- With this feature *enabled*, BRM searches for global charge sharing objects during the discounting process.
- With this feature *disabled*, rating does not search for global charge sharing objects during the discounting process, thereby resulting in better performance.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsRating bus_params_rating.xml
```

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

2. Search the XML file for following line:

   ```
   <EnableGlobalChargeSharing>disabled</EnableGlobalChargeSharing>
   ```

3. Change **disabled** to **enabled**:

   Caution: BRM uses the XML in this file to overwrite the existing **rating** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM rating configuration.

4. Save the file and change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   You should execute this command from the *BRM_Home***/sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using testnap" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.
7. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Using Third-Party Client Applications to Create, Modify, and Delete Global Charge Sharing Groups

To set up your third-party client application to manage global charge sharing groups, you must customize it to accept the following information and pass it in the input flist to the Subscription Management FM opcodes:

- Name of the global charge sharing group
- Owner of the group
- Members of the group
- The chargeshare associated with the group

To add all accounts in the system as members, you pass a type-only account POID and a NULL service POID in the input flist's PIN_FLD_MEMBERS array. For example:

```
0 PIN_FLD_MEMBERS            ARRAY [0] allocated 2, used 2
1   PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account -1 0
1   PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.0 /service 0 0
```

To add all services of a specific type in the system as members, you pass a type-only account POID and a type-only service POID in the input flist's PIN_FLD_MEMBERS array. For example:

```
0 PIN_FLD_MEMBERS            ARRAY [0] allocated 2, used 2
1   PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account -1 0
1   PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.1 /service/email -1 0
```

Note: In both cases, you pass a type-only account POID in the PIN_ FLD_ACCOUNT_OBJ field. The field cannot be NULL.

You pass the information from your client application to the following Subscription Management FM opcodes:

- To create a global charge sharing group, use PCM_OP_SUBSCRIPTION_ SHARING_GROUP_CREATE.
- To modify a global charge sharing group, use PCM_OP_SUBSCRIPTION_ SHARING_GROUP_MODIFY.
- To delete a global charge sharing group, use PCM_OP_SUBSCRIPTION_ SHARING_GROUP_DELETE.

Note: For global charge sharing groups, *do not* call the PCM_OP_ SUBSCRIPTION_ORDERED_BALGRP opcode.

For more information about these opcodes, see "Managing Resource Sharing Groups".

About Creating, Deleting, and Modifying Resource Sharing Groups

You create, delete, and modify a charge sharing group in a similar manner to the way you perform these tasks for a discount sharing group.

Creating Resource Sharing Groups

To create a resource sharing group, you specify a group owner account or service, member services, and the list of group owner's resources that will be shared by the members.

- For discount sharing, the list consists of the discount objects purchased by the group owner.
- For charge sharing, the list consists of all the chargeshares in the database.

See "Setting Up Discount Sharing" or "Setting Up Charge Sharing" in *BRM Configuring Pipeline Rating and Discounting*.

Be aware of the following restrictions:

- Discounts or charges selected for resource sharing must be valid. A valid discount or charge is one that has an active or inactive status and has not expired.

- The owner of a resource sharing group cannot also belong to a member-owned resource sharing group of the same type.

  For example, Anna is the owner of discount sharing group DG1 and wants Sam to be a member. But, Sam is the owner of a different discount sharing group, DG2. Anna can have Sam as a member of DG1 *only* if she is not a member of DG2.
- The name of the resource sharing group must be unique and cannot be used for another resource sharing group owned by the group owner.
- When adding a service as a member, you can add the service as either a service type or a specific service instance.

  If you add a service type as a member, all instances of that service type become members. However, the subtypes of the service type do not become members. For example, if you specify the GSM service type as a member, all instances of GSM become members, but GSM fax, GSM telephony, and so forth do not.
- All members must have the same primary currency as the owner.

Note: This restriction does not apply to global charge sharing groups.

After you create a resource sharing group, members can join the group. When they join the group, an ordered balance group is created. The ordered balance group determines which resource sharing groups the member can use and the order in which the groups are used. See "How Discounts and Charges Are Applied".

You create resource sharing groups in Customer Center. For more information, see the Customer Center Help.

If you are implementing resource sharing through a third-party client application, see "Creating a Resource Sharing Group".

When creating a discount sharing group, the moment when sharing starts is determined by a **pin.conf** entry. See "Configuring the Start and End Times for Discount Sharing".

Deleting Resource Sharing Groups

A resource sharing group is deleted when the group's owner account is closed or when the sharing group is deleted by a CSR. When a discount sharing group is deleted, members can no longer use the discounts that had been shared by the owner. When a charge sharing group is deleted, members charges are no longer assumed by the owner.

When a resource sharing group is deleted, BRM also deletes the resource sharing group from each member's ordered balance group; in effect ending the membership. For more information about ordered balance groups, see "How Discounts and Charges Are Applied".

You delete resource sharing groups in Customer Center. For more information, see the Customer Center Help.

If you are implementing resource sharing through a third-party client application, see "Deleting a Resource Sharing Group".

When deleting a discount sharing group, the moment when sharing ends is determined by a **pin.conf** entry. See "Configuring the Start and End Times for Discount Sharing".

Modifying Resource Sharing Groups

You can modify a resource sharing group in the following ways:

- **Delete members from the group.**

  When a member is deleted from the group, the member can no longer consume resources from the shared resource pool. Deleting a member removes the member from the group's members list. It also removes the group from the member's ordered balance group list. See "How Discounts and Charges Are Applied".

- **Delete shared discounts or sponsored charges.**

  When a shared discount is deleted, members can no longer apply that discount and any owner resource pools for that discount cannot be touched by the members. When a shared charge is deleted, the owning balance group no longer receives balance impacts for member-generated events that are part of that charge.

  Deleting a shared discount or charge removes it from the group's discounts or sponsors list. It also removes the group from the members' ordered balance group lists.

- **Add members to the group.**

  When a member is added to the group, that member can begin benefiting from resource sharing by joining the group. Adding new members updates the group's members list.

  After you add members to the group, they can join the group. When they join the group, an ordered balance group is created and sharing for that member can begin. See "How Discounts and Charges Are Applied".

- **Add shared discounts or charges.**

  When discounts are added to a discount sharing group, members can begin to use the discounts for the events generated by their accounts. When charges are added to a charge sharing group, the owner account begins to receive the balance impact of those charges from the member services.

  Adding new discounts or charges updates the group's discounts or sponsors list. The discounts must be owned by the group owner, and the charges must be defined as chargeshares in the database. The discounts and charges must have an active or inactive status and have valid dates.

- **Change the owner of the resource sharing group.**

  When the owner of a discount sharing group changes, members use discounts from the new owner's shared pool rather than from the old owner's pool. When the owner of a charge sharing group changes, the new owner assumes charges generated by the members and the old owner stops assuming the charges.

  Changing the owner assigns a new group owner, deletes the discounts or sponsors list belonging to the previous owner, and creates a new discounts or sponsors list containing the discounts or charges shared by the new owner.

You modify resource sharing groups in Customer Center. For more information, see the Customer Center Help.

Note: You cannot change the owner of a discount sharing group in Customer Center.

If you are implementing discount sharing through a third-party client application, you customize the application to invoke BRM opcodes. See the following sections:

- Deleting a Resource Sharing Group
- Modifying a Resource Sharing Group
- Changing the Owner of a Resource Sharing Group

When adding or deleting members or discounts in a discount sharing group, the moment when sharing starts and ends is determined by a **pin.conf** entry. See "Configuring the Start and End Times for Discount Sharing".

How Discounts and Charges Are Applied

An account can purchase discounts, share discounts, and have charges that are sponsored. For events generated by a member service, the shared discounts are used before the sponsored charges are applied to the owning balance group. This way, sponsored charges are applied after usage charges have been discounted. To control the sequence in which shared discounts and charges are applied for member-generated events, BRM maintains an ordered balance group for each member.

About Ordered Balance Groups

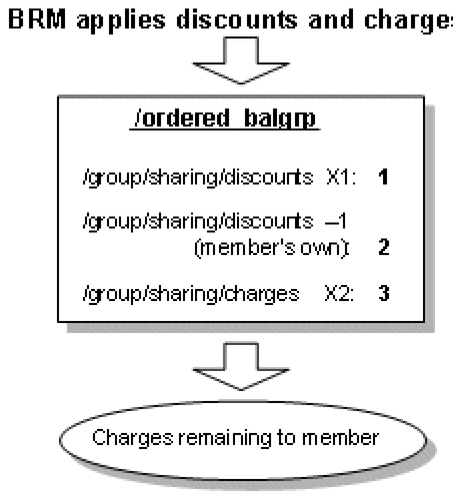
Each member of a discount sharing or charge sharing group has an ordered balance group (**/ordered_balgrp** object). Ordered balance groups are created when members first join resource sharing groups and are updated each time members join new resource sharing groups or leave resource sharing groups.

An ordered balance group contains links to the groups that the member has joined, listed in order by rank. The ordered list determines the sequence in which the group's resource balances are impacted by the member-generated events.

Note: Ordered balance groups do not include any global charge sharing groups to which the member belongs. For more information, see "About Global Charge Sharing Groups".

The following example, as illustrated in Figure 4–7, shows how BRM applies discounts and charges for a member's ordered balance group that references discount sharing group X1 and charge sharing group X2.

Figure 4–7 BRM Application of Discounts and Charges to Ordered Balance Group


BRM applies discounts and charges
/ordered_balgrp
/group/sharing/discounts X1: 1
/group/sharing/discounts –1 (member's own): 2
/group/sharing/charges X2: 3
Charges remaining to member

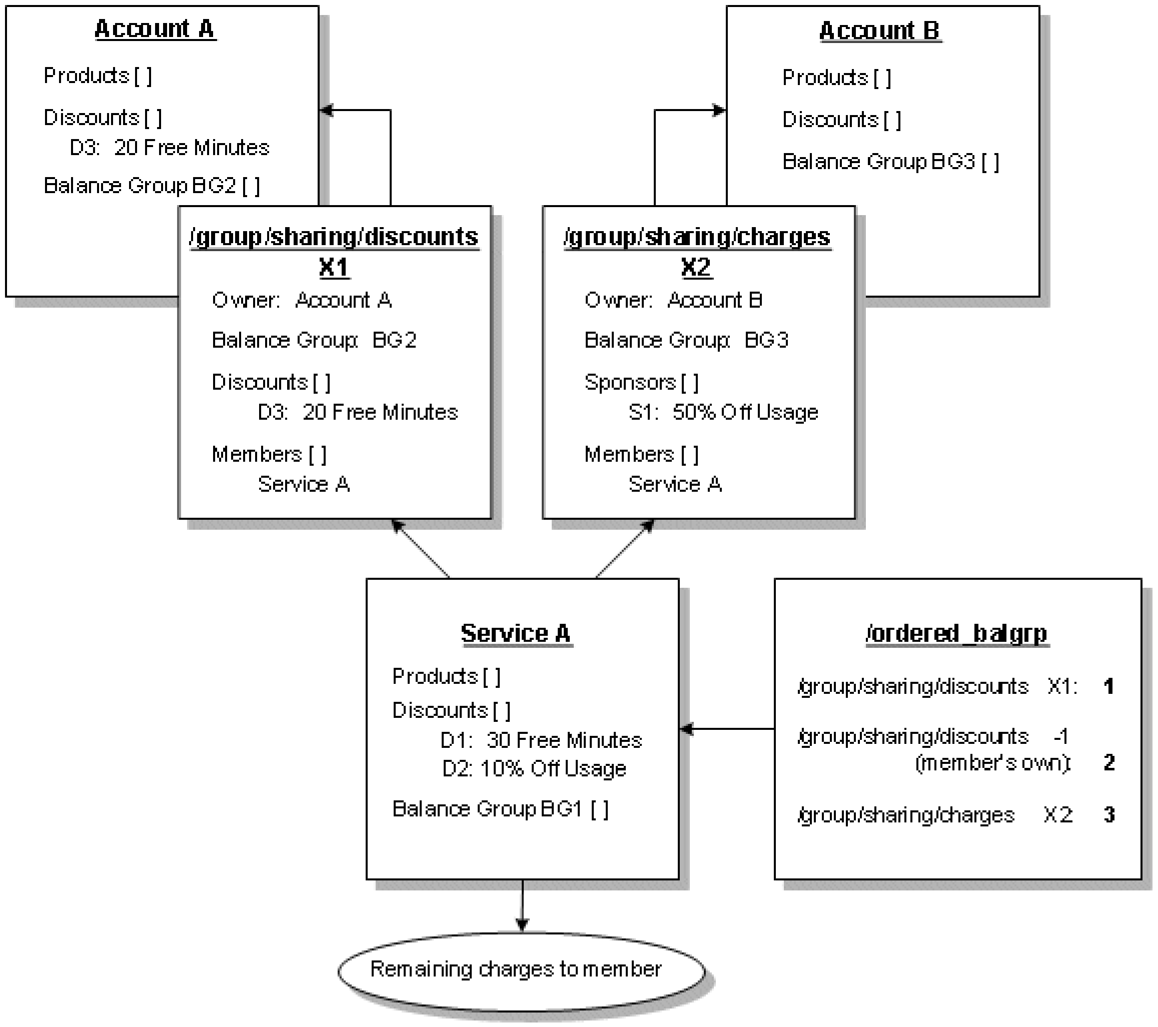
Note: In this example, the entry indicated by **-1** references discounts owned by the member.

As shown in Figure 4–7, when the ordered balance group contains *both* discount and charge sharing groups, the discount sharing group's resources are used first. This enables discounts to be applied before usage charges are determined for charge sharing. If multiple discount or charge sharing groups are configured, they are applied in the order of their rank.

Also, BRM *always* references any charges owned by the member after referencing the ordered balance group. That way, members receive charge sharing benefits before having to use their own charges.

The following example, as illustrated in Figure 4–8, shows how a member's ordered balance group is used to impact the resource balances of the resource sharing groups. In this example, Account A is the owner of discount sharing group X1. Account B is the owner of charge sharing group X2. Service A is a member of both X1 and X2. Account A shares a discount that grants 20 free minutes of telephone calls monthly. Account B sponsors 50% of usage charges. Service A has two discounts, one for 30 free minutes monthly and another that discounts 10% of the charges.

Figure 4–8 Ordered Balance Group Impact on Resource Sharing Groups



Here, BRM applies shared discounts and charges for events generated by Service A as follows:

1. Service A generates an event for 100 minutes of telephone calls at $0.10 per minute.

   Before any discounts or sponsored charges are applied, Service A owes $10.00.

2. Account A's discounts are consumed first.

   After consuming 20 free minutes, Service A owes $8.00 for 80 minutes of telephone calls at $0.10 per minute.

3. Service A's discounts are consumed next.

   After consuming 30 free minutes, Service A owes $5.00 for 50 minutes of calls at $0.10 per minute. After the 10% usage discount, Service A owes $4.50 for 50 minutes of calls.

4. Account B's charges are consumed last.

   After a 50% of usage sponsored by Account B, Account B owes $2.25 and Service A owes $2.25.

How Discounts Are Applied When a Member Belongs to the Group Owner Account

When a member of a discount sharing group belongs to the discount sharing group owner account, the discount at the account level is applied twice for the member.

For example, Account A is the owner of discount sharing group X. Account A owns Service A and Service A is also a member of the discount sharing group X. Account A shares a discount that grants 20 free minutes. When Service A generates an event for 100 minutes, the event is discounted as follows:

- Service A is granted 20 free minutes as a member of the discount sharing group X.
- Because Service A also belongs to the account and any discount purchased at the account level applies to its services, the account-level discount is applied and Service A is granted an additional 20 free minutes.

Creating, Deleting, and Modifying Ordered Balance Groups

Ordered balance groups are created for a member service that participates in resource sharing. A member account can have multiple ordered balance groups depending on how many services are included in resource sharing groups. For example, if a member account participates in resource sharing for its IP, email, and GSM services, it has ordered balance groups for each of these services.

Creating Ordered Balance Groups

To create an ordered balance group, you specify the discount and charge sharing groups that provide resource pools for the service or account.

You create ordered balance groups in Customer Center when a member first joins resource sharing groups for a service. See the Customer Center Help.

If you are implementing resource sharing through a third-party client application, see "Creating an Ordered Balance Group".

Deleting Ordered Balance Groups

To delete an ordered balance group, you specify the ordered balance group that you want to remove from the database.

You cannot delete an ordered balance group in Customer Center. But, when a member decides to end membership in all resource sharing groups for a service, Customer Center deletes all the groups from the ordered balance group list, leaving only the member’s own discounts. See the Customer Center Help.

You can delete an ordered balance group if you are implementing resource sharing through a third-party client application. See "Deleting an Ordered Balance Group".

Modifying Ordered Balance Groups

Modifying an ordered balance group consists of adding or deleting resource sharing groups:

- **Adding a resource sharing group**: A link to the resource sharing group is added to the ordered balance group list.
- **Deleting a resource sharing group**: The link to the resource sharing group is removed from the ordered balance group list.

Important: If you add or delete resource sharing groups, you may need to rearrange the ordered balance group list. See "Modifying the Order in Which Resource Sharing Groups Are Used".

You modify ordered balance groups in Customer Center when a member decides to join a resource sharing group or end membership in a resource sharing group. For information on joining a resource sharing group or on ending membership in a resource sharing group, see the Customer Center Help.

If you are implementing resource sharing through a third-party client application, see "Managing Ordered Balance Groups".

Modifying the Order in Which Resource Sharing Groups Are Used

The sequence of the ordered balance group list determines which shared resource pools will be used first. You can change the order of this list to reprioritize resource sharing for a member.

You modify the order of the ordered balance group list in Customer Center when you join resource sharing groups or if a member decides to use one resource sharing group before another. See the Customer Center Help.

If you are implementing resource sharing through a third-party client application, see "Modifying the Order in Which Sharing Is Applied".

Creating or Modifying Multiple Ordered Balance Groups Simultaneously

To create or modify multiple ordered balance groups simultaneously, you specify the discount and charge sharing group that provides the resource pool. You also specify a list of member services you want to have share this resource pool.

You can create or modify multiple ordered balance groups simultaneously in Customer Center provided it is configured to enable automatic ordered balance group creation. By default, an owner creates a group and adds members to the group. Members, in turn, must explicitly join the group. If, instead, you configure Customer Center to automatically add members, the need for members to explicitly join the group is eliminated. See "Sharing Configurator" in *BRM Developer's Guide*.

If you are implementing resource sharing through a third-party client application, see "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Managing Resource Sharing Groups

There are two types of resource sharing groups; discount sharing groups and charge sharing groups. For general information about discount sharing or charge sharing groups, see "About Resource Sharing Groups".

To manage charge and discount sharing groups, use the following opcodes:

- PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE

  See "Creating a Resource Sharing Group".

- PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY

  See "Modifying a Resource Sharing Group".

- PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE

  See "Deleting a Resource Sharing Group".

- PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT

  See "Changing the Owner of a Resource Sharing Group".
- PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS

  See "Getting a List of Charges Available for Charge Sharing".
- PCM_OP_SUBSCRIPTION_ORDERED_BALGRP

  See "Managing Ordered Balance Groups".
- PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY

  See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Creating a Resource Sharing Group

For general information about discount sharing and charge sharing groups, see "About Resource Sharing Groups".

To create a discount or charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE. Customer Center calls this opcode when you create a sharing group.

The members of a charge sharing group can include individual services, all services of a specific type, or all accounts in the system.

Observe the following guidelines when creating sharing groups:

- Discounts and charges that are shared by the group owner must be valid. A valid discount is one that has an active or inactive status and has not expired.
- If a member owns another sharing group, the owner of the sharing group you create cannot belong to the member's group.
- The name of the resource sharing group must be unique and cannot be used for another resource sharing group owned by the group owner.
- If the member is a service, the service can be either a service type or a specific service instance. When adding a type-only service, all instances of that service type become members. However, the subtypes of the service type do not become members.

If successful, this opcode returns the POID of the resource sharing group object created and the POID of the event generated to record the creation.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE fails in the following cases:

- If the owner is also a member of a sharing group owed by one of the members.
- If a charge or discount in the input flist is not valid.
- If the primary currency for any of the members is different from the owner's primary currency.

Creating a discount or charge sharing group is the first step toward activating discount or charge sharing for the group. For BRM to apply shared discounts and charges for the group's members, you must also create ordered balance groups for the members. See "Creating an Ordered Balance Group".

Note: When creating a global charge sharing group, you do not create or modify each member's ordered balance group.

Creating a Discount Sharing Group

To create a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE:

1. Creates a **/group/sharing/discounts** object and assigns the group owner.

   Note:

   - The owner cannot be a member of its own group.
   - If a member of the group owns a resource sharing group, the owner cannot be a member of that group.

2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/discounts** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the discount sharing group. However, subclass instances of the service type do not become members.

   For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so forth.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/discounts** object is created. The member account can join the group and purchase the service later. BRM begins discount sharing for the service as soon as the member buys the service and joins the discount sharing group. See "Members and Discount Sharing Groups".

3. Adds the **/discount** objects in the PIN_FLD_DISCOUNTS array to the DISCOUNTS array of the **/group/sharing/discounts** object.

   The discounts must be owned by the discount sharing group owner. They must also have validity dates and a valid status (active or inactive). If the PIN_FLD_DISCOUNTS array is empty, all of the owner's discounts are shared. In this case, the PIN_FLD_DISCOUNTS array in the **/group/sharing/discounts** object contains **NULL** in the PIN_FLD_DISCOUNT_OBJ and PIN_FLD_OFFERING_OBJ fields.

4. Generates an **/event/billing/group/sharing/discount** event to record the creation of the discount sharing group.

Creating a Charge Sharing Group

To create a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE:

1. Creates a **/group/sharing/charges** object, and assigns the owner to the group.

   Note: The owner cannot be a member of its own group.

2. Adds the **/account** and**/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/charges** object.

Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the charge sharing group. However, subclass instances of the service type do not become members.

For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so forth.

If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/charges** object is created. The member account can join the group and purchase the service later. BRM begins charge sharing for the service as soon as the member buys the service and joins the charge sharing group. BRM handles future services for charge sharing in the same way that it does for discount sharing. See "Members and Discount Sharing Groups".

3. Adds **/sponsor** objects in the PIN_FLD_SPONSORS array to the SPONSORS array of the **/group/sharing/charges** object.
4. Generates an **/event/billing/group/sharing/charges** event to record the creation of the charge sharing group.

Modifying a Resource Sharing Group

For general information about discount sharing or charge sharing groups, see "About Resource Sharing Groups".

To modify a discount or charge sharing group, use PCM_OP_SUBSCRIPTION_ SHARING_GROUP_MODIFY. Customer Center calls this opcode when you modify a resource sharing group and when you add members to a group.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY can perform the following modifications:

- Adding members to an existing group.
- Adding discounts or charges to an existing group.

If successful, it returns the POID of the group that was modified and the POIDs of the events generated by this opcode to record the group modification.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY fails in the following cases:

- If the owner is a member of its own group or a group owned by a group member.
- If a discount object in the input flist is not valid.

Adding a member to a discount or charge sharing group is the first step toward activating discount or charge sharing for the new member. For the new member to benefit from shared discounts and charges, you must also create an ordered balance group for the member. See "Creating an Ordered Balance Group".

Note: You do not need to create ordered balance groups for members of global charge sharing groups.

Adding Members to a Discount or Charge Sharing Group

To add members to a discount or charge sharing group, PCM_OP_SUBSCRIPTION_ SHARING_GROUP_MODIFY does the following:

Caution: The element ID for each member in the input flist must be unique within the membership. If an element ID is already being used by an existing member in the group object you are modifying, the opcode overwrites the existing member. If you do not know the element IDs of each member in the object, you can prevent member loss by making sure the flist includes both existing and new members.

1. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the group object:

   - The **/group/sharing/charges** object for a charge sharing group.
   - The **/group/sharing/discounts** object for a discount sharing group.

   Each PIN_FLD_MEMBERS array element specifies an **/account** and a **/service** object. If the **/service** object is NULL, the **/account** is the member.

   Note:

   - The owner cannot be a member of its own group.
   - If a member of the group owns a resource sharing group, the owner cannot be a member of that group.
   - If the PIN_FLD_MEMBERS array is empty, the opcode calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE to delete all members from the group.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the discount sharing group. However, subclass instances of the service type do not become members.

   For example, if the input flist specifies the GSM service type, the opcode adds all **/service/telco/gsm** instances as members. But, it does not add the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so forth.

   If you specify a service type as a member, the member account does not need to own the service when the group object is modified. The member account can join the group and purchase the service later. BRM begins discount or charge sharing for the service as soon as the member buys the service and joins the discount or charge sharing group. See "Members and Discount Sharing Groups".

2. Generates an event to record member changes:

   - A **/group/sharing/discounts/modify** event for a discount sharing group.
   - A **/group/sharing/charges/modify** event for a charge sharing group.

Adding Discounts to a Discount Sharing Group

To add discounts for a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/discount** objects in the PIN_FLD_DISCOUNTS array to the **/group/sharing/discounts** object's discounts array.

Note:

- The discount instance must be owned by the group owner. They must also have validity dates and a valid status (active or inactive).
- If the PIN_FLD_DISCOUNTS array is empty, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE is called to delete all **/discount** objects from the **/group/sharing/discounts** object's discounts array.

---

2. Generates a **/group/sharing/discounts/modify** event to record the event.

Adding Sponsored Charges to a Charge Sharing Group

To add sponsored charges to a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/sponsorship** objects in the PIN_FLD_SPONSORS array to the **/group/sharing/charges** object's sponsors array.

Note: If the PIN_FLD_SPONSORS array is empty, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE is called to delete all **/sponsor** objects from the **/group/sharing/charges** object's SPONSORS array.

---

2. Generates a **/group/sharing/charges/modify** event to record the event.

Deleting a Resource Sharing Group

For general information about discount sharing or charge sharing groups, see "About Resource Sharing Groups".

To delete a discount or charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE. You can also use this opcode to delete shared discounts, sponsored charges, and group members.

Customer Center calls this opcode when:

- You delete a resource sharing group.
- You delete members from a group.
- You delete discounts or charges from the resource sharing group.

Deleting a Discount Sharing Group

To delete a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. For each member in the group, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **group/sharing/discounts** object from the member's **/ordered_balgrp** object.
2. Deletes the **group/sharing/discounts** object.
3. Generates an **/event/billing/group/sharing/discounts/delete** event to record the deletion.

Deleting a Charge Sharing Group

To delete a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. For each member in the group, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **group/sharing/charges** object from the member's **/ordered_balgrp** object.
2. Deletes the **/group/sharing/charges** object.
3. Generates an **/event/billing/group/sharing/charges/delete** event to record the deletion.

Deleting a Member from a Discount Sharing Group

To delete a member from a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

Caution: The element ID for each member in the **/group/sharing/discounts** object is unique. If you use an incorrect element ID for the member you want to delete, the opcode deletes the wrong member.

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **group/sharing/discounts** object from the member's **/ordered_balgrp** object.
2. Deletes the member from the **/group/sharing/discounts** members array.
3. Generates an **/event/billing/group/sharing/discounts/delete** event.

Deleting a Member from a Charge Sharing Group

To delete a member from a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

Caution: The element ID for each member in the **/group/sharing/charges** object is unique. If you use an incorrect element ID for the member you want to delete, you will delete the wrong member.

1. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to delete the **/group/sharing/charges** object from the member's **/ordered_balgrp** object.
2. Deletes the member from the **/group/sharing/charges** members array.
3. Generates an **/event/billing/group/sharing/charges/delete** event.

Deleting a Shared Discount from a Discount Sharing Group

To delete a shared discount from the discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

Caution: The OFFERING_OBJ uniquely identifies each shared discount in the **/group/sharing/discounts** object. If you use an incorrect OFFERING_OBJ for the discount you want to delete, you will delete the wrong discount.

1. Deletes the **/discount** objects specified in the PIN_FLD_DISCOUNTS array from the DISCOUNTS array of the **/group/sharing/discounts** object.
2. Generates an **/event/billing/group/sharing/discounts/delete** event to record the deletion.

Deleting a Sponsored Charge from a Charge Sharing Group

To delete a sponsored charge from the charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

Caution: The element ID for each shared charge in the **/group/sharing/charges** object is unique. If you use an incorrect element ID for the charge you want to delete, you will delete the wrong charge.

1. Deletes the **/sponsor** objects specified in the PIN_FLD_SPONSORS array from the SPONSORS array of the **/group/sharing/charges** object.
2. Generates an **/event/billing/group/sharing/charges/delete** event to record the deletion.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE returns the POID of the resource sharing group object that was modified and the POID of the event that was generated.

Changing the Owner of a Resource Sharing Group

For general information about discount sharing or charge sharing groups, see "About Resource Sharing Groups".

To change the owner of a discount or charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT. This opcode takes as input:

- PIN_FLD_GROUP_OBJ: The sharing group (**/group/sharing/discounts** or **/group/sharing/charges**) whose owner is being changed.
- PIN_FLD_POID: The **/account** object of the current owner.
- PIN_FLD_PARENT: The new owner. This field identifies the **/service** object designated as the new owner of the resource sharing group. The **/service** object can be in the account that currently owns the resource sharing group or it can be in a different account.
- PIN_FLD_ACCOUNT_OBJ: The **/account** object of the new owner.

Note: If you are changing the ownership of the resource sharing group from one service to another within the same account, the object passed in this field matches the one in the PIN_FLD_POID field.

- PIN_FLD_BAL_GRP_OBJ: The balance group used to track sharing activities for the new owner.
- PIN_FLD_DISCOUNTS: An array of discounts that the new owner will share with the group members. This field is optional but should be included if the shared discounts for the new owning service are different from those for the current owning service.

Note: Include this array only if you are using the opcode to change the owner of a discount sharing group.

If successful, this opcode returns the POID of the resource sharing group that was modified and the event that was generated to record the owner change.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT fails in the following cases:

- If the new owner that is passed is a member of the resource sharing group.

This opcode can also be used to change the owner of a balance monitor, provided you are changing the owner manually and not through Automated Monitor Setup (AMS). For details on balance monitoring and AMS, see "About Balance Monitoring". For information on changing the owner of a balance monitor, see "Changing the Owner of a Balance Monitor".

Changing the Owner of a Discount Sharing Group

To change the owner of a discount sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - The input flist contains the **/balance_group** object and **/account** object for the new discount sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - The currency of the new owner is the same as the currency of the old owner.
   - There are no circular relationships. For example, the owner cannot be a member of its own group. Also, if a member of the group owns a resource sharing group, the owner cannot be a member of that group.
2. Deletes the current owner's discounts from the **/group/sharing/discounts** object.
3. Sets the **/group/sharing/discounts** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
4. Adds the new owner's shared discounts to the **/group/sharing/discounts** object.
5. Generates an **/event/group/sharing/discount/modify** event to record the owner change.

Changing the Owner of a Charge Sharing Group

To change the owner of a charge sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - The input flist contains the **/balance_group** object and **/account** object for the new charge sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - The currency of the new owner is the same as the currency of the old owner.
   - There are no circular relationships. For example, the owner cannot be a member of its own group.
2. Sets the **/group/sharing/charges** object's owner to the new owner specified in the PIN_FLD_PARENT input field.

3. Updates the sponsor flags in the old owner's and new owner's **/billinfo** objects.
4. Generates an **/event/group/sharing/charges/modify** event to record the owner change.

Getting a List of Charges Available for Charge Sharing

To get a list of all charges available for charge sharing, use PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS.

When you create a charge sharing group, you select the charges, or chargeshares, that you want the owner to assume for the members. Chargeshares are stored in **/sponsorship** objects in the BRM database. By default, this opcode retrieves all the **/sponsorship** objects.

You can customize this opcode to filter or return a subset of **/sponsorship** objects from the list of all available **/sponsorship** objects. For example, you can customize the opcode to get a list of the chargeshares that include a particular set of events.

Managing Ordered Balance Groups

For general information about ordered balance groups, see "About Ordered Balance Groups".

To create, modify, or delete an ordered balance group (**/ordered_balgrp** object) for an account or service, use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP. Customer Center calls this opcode whenever a member joins or leaves a resource sharing group.

Note: PCM_OP_SUBSCRIPTION_ORDERED_BALGRP lets you create or modify one ordered balance group at a time. To create or modify more than one ordered balance group at time, use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

An ordered balance group object is created when an account or service joins a resource sharing group. You can modify an ordered balance group by:

- Adding a resource sharing group.
- Deleting a resource sharing group.

The object stores the sharing groups that the member has joined in its ORDERED_BALGRPS array so that balance impacts are applied to the owning balance group for each sharing group. If the array contains no resource sharing groups, all balance impacts are applied to the member's balance group.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP uses the element IDs in the ORDERED_BALGRPS array to determine the order in which to apply the shared discounts and shared charges. The array has two segments; one for discount sharing groups and one for charge sharing groups. The segment for discount sharing groups is independent of the segment for charge sharing groups; BRM does not intermingle the two groups. Discount sharing groups always precede charge sharing groups within the array.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP takes as input an **/ordered_balgrp** object and a string that specifies whether to create, modify, delete, or list the ordered balance group object.

Note: The **/ordered_balgrp** object must be specified if the action is to modify or delete the object.

If successful, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP returns:

- The POID of the **/ordered_balgrp** object
- The POID of the event generated as a result of the creation, modification, or deletion.

If the action is **List**, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP returns all the resource sharing group objects and their ranks in the **/ordered_balgrp** object.

After you have created an **/ordered_balgrp** object, you can modify the sequence of the groups in the ordered balance group list. See "Modifying the Order in Which Sharing Is Applied".

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP fails in the following cases:

- If there is more than one instance of a resource sharing group in the PIN_FLD_ORDERED_BALGROUPS array.
- If the PIN_FLD_ACTION is Create but there is already an **/ordered_balgrp** object for the account or service.

Creating an Ordered Balance Group

To create an ordered balance group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Creates an **/ordered_balgrp** object for the **/account** or **/service** object specified in the input flist.
2. Adds the resource sharing group objects specified in the input flist to the **/ordered_balgrp** object.

   The opcode arranges the sharing groups in the order defined by the element IDs in the input flist. The **/group/sharing/discounts** objects always come before the **/group/sharing/charges** objects so that shared discounts are applied before the shared charges impact the owning balance group.
3. Generates an **/event/billing/ordered_balgrp/create** event to record the object creation.

You can also use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY opcode to create one or more ordered balance groups. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Adding a Resource Sharing Group to an Ordered Balance Group

To add a resource sharing group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Adds the resource sharing group objects specified in the input flist to the **/ordered_balgrp** object.
2. Generates an **/event/billing/ordered_balgrp/modify** event to record the object modification.

Important: You can also use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY opcode to add a resource sharing group to one or more ordered balance groups. See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Deleting a Resource Sharing Group from an Ordered Balance Group

To delete a resource sharing group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

Caution: The element ID for each resource sharing group in the **/ordered_balgrp** object is unique. If you use an incorrect element ID for the resource sharing group you want to delete, you will delete the wrong resource sharing group.

1. Deletes the resource sharing group objects specified in PIN_FLD_ORDERED_BALGROUPS array from the **/ordered_balgrp** object.

   Note: If the action is **Delete** but there is no PIN_FLD_ORDERED_BALGROUPS array in the input flist, the **/ordered_balgrp** object is deleted. See "Deleting an Ordered Balance Group".

2. Generates an **/event/billing/ordered_balgrp/modify** event to record the object modification.

Modifying the Order in Which Sharing Is Applied

To modify the order in which BRM applies shared discounts and shared charges, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Replaces the resource sharing group objects in the **/ordered_balgrp** object with the set of resource sharing group objects specified in the input flist.

   The replacement set of sharing group objects can reference the same resource sharing groups as before, but the element IDs used to establish the order are different.

   Caution: When you use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to re-order the list, it overwrites the **/ordered_balgrp** object, so you must include all the resource sharing groups in the input flist.

2. Generates an **/event/billing/ordered_balgrp/modify** event to record the object modification.

You can also use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY opcode to add a resource sharing group to one or more ordered balance groups.

See "Creating and Modifying Multiple Ordered Balance Groups Simultaneously".

Deleting an Ordered Balance Group

To delete an ordered balance group, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP does the following:

1. Deletes the **/ordered_balgrp** object specified in the input flist.
2. Generates an **/event/billing/ordered_balgrp/delete** event to record the object deletion.

Creating and Modifying Multiple Ordered Balance Groups Simultaneously

You can create multiple ordered balance groups individually or add a resource sharing group to multiple ordered balance groups simultaneously. To do so, use PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY.

Customer Center calls this opcode to automatically add members to a resource sharing group if so configured. See "Sharing Configurator" in *BRM Developer's Guide* and "Creating or Modifying Multiple Ordered Balance Groups Simultaneously".

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY takes as input the POID of the resource sharing group, an array of one or more members, and a field indicating whether to add the resource sharing group to the beginning or end of the appropriate segment in the ordered balance group list (discount sharing segment or charge sharing segment).

If a service identified in the PIN_FLD_MEMBERS array specifies a type-only service, such as GSM, this opcode creates or modifies **/ordered_balgrp** objects for all instances for that service type. However, it does not create or modify **/ordered_balgrp** objects for any of the subclass instances.

For example, if the input flist specifies the GSM service type, the opcode creates or modifies **/ordered_balgrp** objects for all **/service/telco/gsm** instances. But, it does not create or modify **/ordered_balgrp** objects for any the **/service/telco/gsm/data** instances, **/service/telco/gsm/sms** instances, and so forth.

If successful, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY returns the POID of the resource sharing group and an array of the **/event/billing/ordered_balgrp/create** and **/event/billing/ordered_balgrp/modify** events that record the action taken for each ordered balance group in the input flist.

PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY fails in the following cases:

- If there is more than one instance of an ordered balance group in the PIN_FLD_MEMBERS array.
- If the discount or charge sharing group (PIN_FLD_POID) is already included in some existing **/ordered_balgrp** objects in the PIN_FLD_MEMBERS array.

To create or modify ordered balance groups, PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY does the following:

1. Verifies that the input flist contains a PIN_FLD_ORDERED_BALGROUPS array with at least one member.
2. Selects a member in the PIN_FLD_MEMBERS array and determines whether the database already contains an **/ordered_balgrp** object for that member.

   If not, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to create the object. See "Creating an Ordered Balance Group".
3. Selects a member in the PIN_FLD_MEMBERS array and determines whether the database already contains an **/ordered_balgrp** object for that member.

   If not, it calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to create the object. See "Creating an Ordered Balance Group".

4. Calls PCM_OP_SUBSCRIPTION_ORDERED_BALGRP to modify the **/ordered_balgrp** object so that it includes the new resource sharing group.

   See "Adding a Resource Sharing Group to an Ordered Balance Group".

5. Repeats steps 2 through 4 for each remaining **/ordered_balgrp** in the PIN_FLD_MEMBERS array.

5

Creating and Managing Sponsor Groups

This chapter describes how to create and manage sponsored accounts in your Oracle Communications Billing and Revenue Management (BRM) system.

For information about different types of groups, see "About Account Groups".

About Sponsor Groups

Note: Sponsor groups are supported for backward compatibility. If you have not set up sponsorship, create resource sharing groups instead. Pipeline Manager discounting is required to set up resource sharing groups. If you do not use Pipeline Manager, you can configure discounts at the deal-level, however, you will not be able to create the various discount configurations that are available with Pipeline Manager discounting. For information about resource sharing groups, See "About Resource Sharing Groups". For information about Pipeline Manager discounting, see "About Discounts" in *BRM Configuring Pipeline Rating and Discounting*. For information about deal-level discounts, see "Providing Deal-Level Discounts" in *BRM Setting Up Pricing and Rating*.

You manage sponsored billing by setting up sponsor groups. A sponsor group consists of one sponsor group owner account and one or more member accounts. An owner account can sponsor multiple sponsor groups, and a member account can belong to multiple sponsor groups. A member account does not need to use the same billing cycle as the sponsor owner account.

You can use sponsored billing in the following cases:

- **Sponsoring a service**. For example, a subscriber pays for Internet access, and the subscriber's grandmother pays for a gaming service as a gift.
- **Sponsoring part of a service**. For example, a company pays an employee's monthly fee for Internet access, which includes 10 free hours per month. The employee pays an hourly rate for any usage over 10 hours per month.
- **Crediting the sponsoring account for sponsored usage**. For example, a sponsoring company pays an employee's monthly fee for Internet access. For every hour the employee uses, the sponsoring company receives a credit against its cost of providing the monthly service.
- **Crediting one account and charging another account in a single transaction**. For example, if you pay a commission to another company for a service that you

provide, you can use sponsored rates to convert part of a balance impact into a credit that provides the commission.

Figure 5–1 shows a sponsor group whose owner pays for its members Internet access but not for their email.

Figure 5–1 Sponsorship of Internet Access Charges

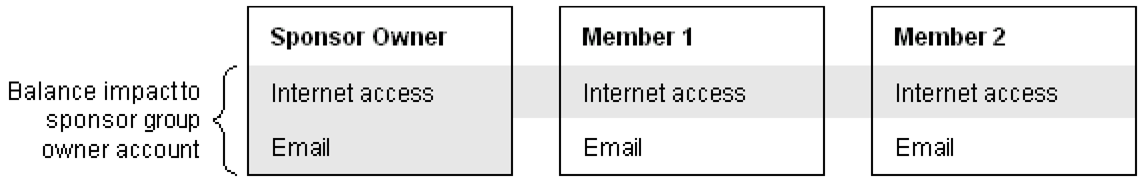
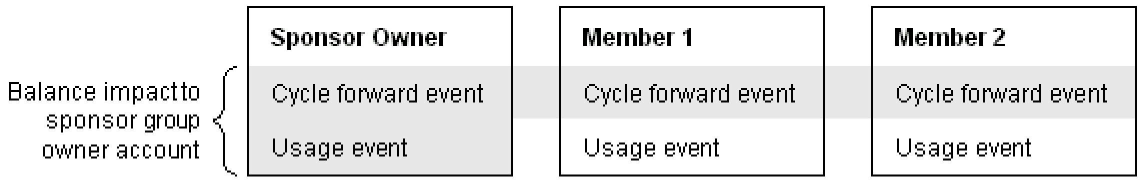


Figure 5–2 shows a sponsor group whose owner pays its members' monthly subscription fees but not their usage fees.

Figure 5–2 Sponsorship of Cycle Forward Event Fees



How Charges Are Sponsored

In Customer Center, you select rate plans for sponsor groups to sponsor. For sponsorship to occur, the appropriate resources in each sponsored rate plan's rates must be set as sponsorable in your company's price list.

For example, sponsor group A sponsors rate plan B from product C, which is associated with an Internet access service. Rate plan B contains three usage rates based on time of day: rate 1 (8 a.m. to 5 p.m.), rate 2 (5 p.m. to 11 p.m.), and rate 3 (11 p.m. to 8 a.m.). Each rate tracks balance impacts in US Dollars and euros. The US Dollar resource is set as sponsorable in rates 2 and 3. In this situation, the owner of sponsor group A pays usage charges for all members of the group whose currency resource is US Dollars when they use the access service any time from 5 p.m. to 8 a.m. The owner of sponsor group A does not pay any charges for members whose currency resource is euros or for any members when they use the service from 8 a.m. to 5 p.m.

If a member account receives charges that should be sponsored, use Pricing Center to verify that the appropriate resources in the rate plan's rates are sponsorable.

For more information about rate plans and rates, see "About Creating a Price List" in *BRM Setting Up Pricing and Rating*.

When Sponsorship Begins

Sponsorship begins when an account becomes a member of a sponsor group. If the account owned products sponsored by the group before the account was added to the group, those products are sponsored as soon as the account joins the group. Charges incurred before the account was added to the group, however, are not retroactively sponsored.

Creating and Managing Sponsored Groups

Use Customer Center to create and manage sponsored groups. For more information, see the Customer Center Help.

Creating Sponsorship Bill Units

By default, an account is created with one bill unit. When you create sponsor groups in Customer Center, the bill units in the accounts are assigned the same sponsor relationships as the accounts to which they belong.

You can create additional bill units for accounts by writing custom code using the BRM API. For more information, see "Managing Bill Units with Your Custom Application" in *BRM Configuring and Running Billing*.

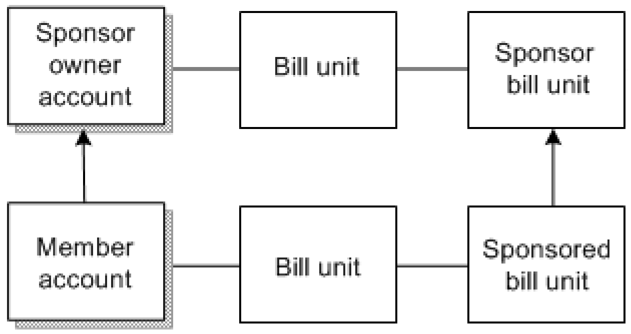
When you create additional bill units, you associate an account and a payment method with the new bill unit (**/billinfo** object). To associate account balances with a bill unit's bill, you must link the account **/balance_group** objects to the new **/billinfo** object. To create sponsorship bill units for accounts that have multiple bill units, you specify the sponsorship relationships in fields in the **/billinfo** objects.

After you have created the bill unit sponsorship associations, you must also pass the appropriate **/billinfo** objects to the opcodes that process billing and payments.

For accounts that have multiple bill units, bill unit sponsorship becomes more complex. Bill units in an account can be sponsor owners or sponsored members. Sponsor owner accounts can have bill units that do not sponsor any charges, and sponsored member accounts can have bill units that do not have any sponsored charges.

In Figure 5–3, the member account has two bill units, but only one of them is handled by the sponsor account.

Figure 5–3 Complex Sponsorship



Guaranteed Sponsorship

You can specify that the sponsor guarantees to pay the sponsored charges, even if the balance of the sponsor's account exceeds the credit limit. Guaranteed sponsorship applies to all the rate plans a group sponsors.

For more information, see the Customer Center Help.

How Account Status Changes Affect Sponsor Groups

When a sponsor group owner account is inactivated, sponsorship is suspended.

When an inactive sponsor owner account is reactivated, it automatically resumes paying its member accounts' sponsored charges.

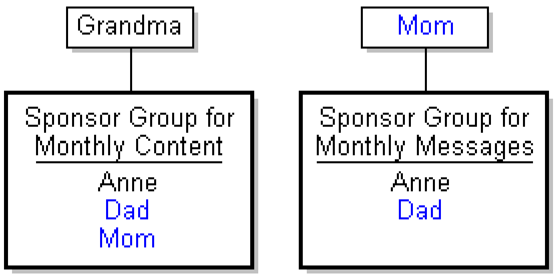
When a sponsor group owner is closed, sponsored charges no longer affect the owner account's balance, and one of the following occurs:

- All members are removed from the owner's sponsor groups, and the groups are deleted. Member accounts begin paying all charges for the formerly sponsored products.
- All members are removed from the owner's sponsor groups, the groups are deleted, and the sponsored products are canceled.
  - This operation can be time-consuming. BRM checks all products owned by each member account, determines whether the products belong to the owner account's sponsor groups, and then cancels those that do.
  - Member accounts pay all cancellation fees for sponsored rate plans.
- All sponsor groups owned by the closed account are assigned to a new sponsor owner account. All groups are assigned to the same new owner. Sponsored fees accrued before the old owner is closed are charged to the old owner. After closure, all fees are charged to the new owner.

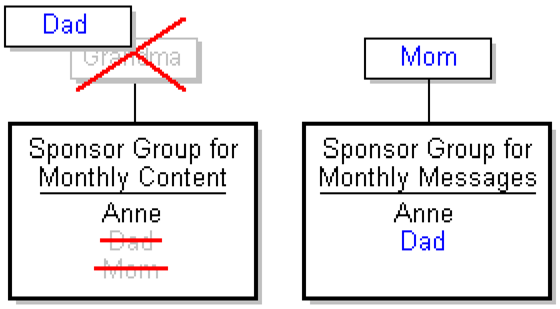
Caution: When groups are assigned to a new owner, the following members (if they exist) are deleted from the groups:

- The new owner account
- Accounts that sponsor the new owner

After being deleted, the preceding accounts begin paying all charges formerly sponsored by the groups. For example, in the following figure, sponsor owner Grandma pays monthly content service charges for members Anne, Dad, and Mom. Sponsor owner Mom pays monthly short message service charges for members Anne and Dad:


Grandma
Sponsor Group for Monthly Content
Anne
Dad
Mom
Mom
Sponsor Group for Monthly Messages
Anne
Dad

If you close Grandma's account and reassign her sponsor group to Dad's account, Dad is deleted from the group because he is the group's new owner. Mom is deleted from the group because Dad is a member of a group she sponsors. Dad and Mom now pay their own content service charges:


Dad
~~Grandma~~
Sponsor Group for Monthly Content
Anne
~~Dad~~
~~Mom~~
Mom
Sponsor Group for Monthly Messages
Anne
Dad

Important: If a member account purchases a sponsored product while owned by account A and then cancels the product while owned by account B, all refunds or cancellation fees are applied to account B. Thus, account B might get a refund for a fee it did not pay, or it might be charged a cancellation fee for a product it did not purchase.

When a closed sponsor owner account is reactivated, former sponsor relationships between owner and member accounts must be manually re-created.

When a member accounts is closed, the cancellation fees for sponsored rate plans are applied to the sponsor owner account, not to the member account.

By default, when a member account is closed, it is removed from all the sponsor groups where it is a member. To specify whether to remove the member account from the sponsor groups, you modify the **billing** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For details, see "pin_bus_params" in *BRM Developer's Guide*.

To specify whether to remove a member account from sponsor groups:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

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

2. Search the XML file for following line:

   ```
   <RemoveSponsoree>disabled</RemoveSponsoree>
   ```

3. Change **disabled** to **enabled**. For this parameter:

   - **disabled** means that BRM keeps the member account in the sponsor groups.
   - **enabled** means that BRM removes the member account from the sponsor groups.

   Caution: BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

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

   ```
   pin_bus_params bus_params_billing.xml
   ```

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

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using testnap" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

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

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

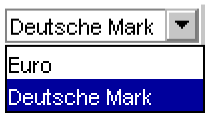
   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Currency Requirements of Sponsor Groups

The sponsor group owner account and all accounts it sponsors must use the same currency. If the accounts use two currencies, they must use the same primary account currency.

To determine the currency or currencies an account uses, in Customer Center, open the account and look on the toolbar. The first currency in the selector list is the primary currency as shown in Figure 5–4:

Figure 5–4 Primary Currency in Customer Center



Product Ownership Requirements of Sponsor Groups

The sponsor group owner account does not need to own the sponsored products.

Brand Requirements of Sponsor Groups

If your company supports Branded service management, group owners and members must belong to the same brand.

Troubleshooting Sponsorship

- The resources in all rates in sponsored rate plans should be set as sponsorable in your company's price list. If a member account receives charges associated with a sponsored rate plan, use Pricing Center to verify that all the rate plan's resources are sponsorable.
- Your billing setup must be configured to bill member accounts before owner accounts. If it is not, the members' sponsored charges might not be included in the owner's bill for the current billing cycle. Instead, they are added to the owner's bill for the next billing cycle.

  See "Setting Up Billing for Sponsorship" in *BRM Configuring and Running Billing*.
- Each product in a member account should have only one sponsor.

  See "One Sponsor per Product" in *BRM Setting Up Pricing and Rating*.
- Sponsorship is suspended when the status of the sponsor owner account is inactive.
- Sponsorship may end when the status of the sponsor owner account is closed.

  See "How Account Status Changes Affect Sponsor Groups".

Managing Sponsor Groups in Your Custom Application

For information about sponsored billing, see "About Sponsor Groups".

Use the following opcodes to manage sponsor groups:

- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_CREATE

  See "Creating a Sponsor Group".
- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_MODIFY

  See "Modifying a Sponsor Group".
- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT

  See "Setting the Parent of a Sponsor Group".

- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER

  See "Adding a Member to a Sponsor Group".

- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER

  See "Deleting a Member from a Sponsor Group".

- PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE

  See "Deleting a Sponsor Group".

Creating a Sponsor Group

To create a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_CREATE. This opcode takes the POID of the sponsor's account as input and performs the following functions:

- Creates the **/group/sponsor** object
- Initializes the sponsor group object by using the name for the sponsor group specified in the input flist
- Initializes the sponsor group object with the sponsored product and rate information, if specified in the input flist
- Creates an **/event/group/parent** object to record the creation of the sponsor group

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_CREATE 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 sponsor group.

Modifying a Sponsor Group

To modify a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_MODIFY.

This opcode takes the POID of the sponsor group and sponsored product and rate information as input and updates the sponsor group object by adding or removing the product and rate specified.

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_MODIFY returns the POID of the **/group/sponsor** object passed in the input flist.

Setting the Parent of a Sponsor Group

To set the parent of a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT.

This opcode takes the account POID of the new sponsor group owner and the POID of the sponsor group as input, and performs the following functions:

- Verifies that the currencies of the existing sponsor group owner and the proposed new sponsor group owner are the same.
- Checks whether any members of the sponsor group are already sponsoring the proposed new sponsor group owner in another group.

Important: A sponsor group owner cannot, be sponsored by a member of its sponsor group. If any of the existing members of the sponsor group form such a cyclic relationship with the proposed new sponsor group owner, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT drops the member from the **/group/sponsor** object.

- Sets the sponsor group owner field of the **/group/sponsor** object.

  ```
  /parent 204438794022169271 0
  ```

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT returns the POID of the event **/group/member/parent** that is created.

If the currency of the new sponsor parent does not match the currency of the existing sponsor parent, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT logs an error in the CM **pinlog** file with the error code PIN_ERR_CURRENCY_MISMATCH.

If any members of the sponsor group form a cyclic relationship with the new sponsor group owner (that is, if any members are sponsoring the new sponsor group owner in another group), PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_SET_PARENT logs a message in the CM **pinlog** file that indicate the POIDs of those members.

Adding a Member to a Sponsor Group

To add a member to a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER.

This opcode takes the POID of the sponsor group object and the POID of the member's account object as input and performs the following operations:

- Verifies that the member account uses the same currency as the sponsor group owner account.

  Important: The PIN_FLD_CURRENCY field in the input flist is optional. Set PIN_FLD_CURRENCY to specify the currency for the member and for the sponsor owner. If currency information is not provided in the input flist, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER queries the database for this information; as a result, the opcode takes slightly longer to finish.

- Verifies that the member account does not have the sponsor owner account as its member in a cyclic relationship. For example, where account A sponsors account B, and Account B attempts to sponsor account A. This is a cyclic relation.
- Adds the member to the sponsor group.
- Creates an **/event/group/member** object to record the addition of a sponsor group member.

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER 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 sponsor group

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_ADD_MEMBER fails in the following cases:

- If the member has the same currency as the sponsor
- If a cyclic relation exists between the member account and the sponsor account

Deleting a Member from a Sponsor Group

To delete a member from a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER.

This opcode takes the POID of the sponsor group and the POID of the member's account as input, and performs the following functions:

- Verifies if the member is sponsored by the sponsor group
- Removes the member from the sponsor group
- Creates an **/event/group/member** object to record the deletion of the sponsor group member

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER 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 sponsor group member

PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE_MEMBER fails if the member is not sponsored by the sponsor group passed in the input flist.

Deleting a Sponsor Group

To delete a sponsor group, use PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE.

This opcode takes the POID of the sponsor group as input and performs these functions:

- Deletes the sponsored accounts from the sponsor group
- Deletes the **/group/sponsor** object
- Creates an **/event/group/member** object to record the deletion of the sponsor group

If successful, PCM_OP_SUBSCRIPTION_SPONSOR_GROUP_DELETE returns the POID of the **/group/sponsor** object that was passed in the input flist.

6 Configuring Adjustments, Disputes, and Settlements

This chapter provides an overview of how adjustments, disputes, and settlements are performed in Oracle Communications Billing and Revenue Management (BRM). It also tells you how to configure taxes, resource reservation, and reason codes for this functionality.

For more information on adjustments, disputes, and settlements, see the following topics:

- About Accounts Receivable
- Using the Event Browser to Display and Adjust Events

For information on creating adjustments, disputes, and settlements in Customer Center, see the Customer Center Help.

About Adjustments

An adjustment is a transaction that debits or credits a customer's account by changing the amount due for a bill item. Adjustments can also change non-currency balances in the account. For information on currency and non-currency adjustments, see "About Adjustments".

Customer service representatives perform adjustments on a variety of levels, as appropriate to the situation. For instance, if the customer made a 10-minute call that was mistakenly billed as a 30-minute call, CSRs perform the adjustment for that specific call at the event level. If, however, the customer's plan provided 100 free minutes a month, but charges started accruing after only 30 minutes, CSRs perform the adjustment at the account level instead.

The way that BRM processes adjustments and records the adjustment's balance impact varies slightly from level to level, as follows:

- **A/R and individual account level**: Adjustments at this level reduce the current balance of the customer's bill. The account's default balance group will be decreased by the amount of the credit.
- **Subscription service level and member service level**: Adjustments at these levels are similar to account-level adjustments. However, the adjustment targets a specific balance group associated with the subscription service or member service rather than using only the default balance group. In this case, BRM uses the balance group supplied by the **/service** object associated with the subscription service or member service selected by the CSR.

As with account-level adjustments, the CSR must allocate the adjustment before it affects the customer's bill.

- **Bill level**: Like the higher-level adjustments, adjustments at this level reduce the current balance of the customer's bill. Here, the amount of the adjustment is subtracted from the due amount for the bill, and payment for that amount is not requested.

  Adjustments can be made to an entire bill or a selection of bill items, distributing the adjustment as a fixed amount per item or as a percentage. In either case, BRM creates a single adjustment item and transfers the credit to the bill items covered by the adjustment. The current balance of the appropriate balance groups is reduced by the amount of the credit.

  Bill-level adjustments only act against A/R bills. CSRs cannot adjust a bill from a subordinate bill unit by filing a bill-level adjustment directly against that bill. Instead, they file the bill-level adjustment against the parent A/R bill. Also, the adjustment amount cannot exceed the total amount of the bill against which the adjustment is applied.

- **Item level**: When you adjust a bill item, the amount of the adjustment is subtracted from the Due of the bill item, and payment for that amount is not requested. The current balance of the appropriate balance group is reduced by the amount of the credit.

- **Event level**: Adjustments at this level depend on whether the adjustment occurs before billing or after billing. In either case, the original event is never adjusted.

  – If the adjustment occurs before billing has run, it changes the balance impact of the shadow event.

  – If the adjustment occurs after billing has run, it changes the balance impact of the adjustment event (**/event/billing/adjustment/event**).

  An event adjustment that credits currency reduces the balance impact of the event and its General Ledger (G/L) impact. It does not cancel the event, only the cost of the event.

  > **Important:** When adjusting pending items, ensure proper G/L reporting by specifying that BRM create a shadow adjustment instead of a standard adjustment. See "Categorizing Unbilled Event-Level Adjustments in G/L Reports".

  CSRs can adjust multiple events in one operation. BRM creates one adjustment item for the adjustments performed in an operation and transfers the credit to all the bill items associated with the adjusted events.

  CSRs can also adjust a given event multiple times. For example, a customer indicates an evening event was billed at a higher rate than it should have been and the CSR adjusts the event to change the rate from a day rate to an off-hours rate. Later, the CSR realizes that a holiday rate should have been applied instead and performs a second adjustment to switch to the holiday rate.

BRM performs adjustments at each of these levels by calling different adjustment opcodes. For information on how to customize adjustments and on which opcodes to call from your client application, see "Performing Disputes and Settlements with Your Custom Application".

CSRs make adjustments on all levels by using Customer Center. For the steps to perform adjustments in Customer Center, see "About Adjustments".

Note: Although Customer Center provides the easiest method of adjusting events, CSRs can also make event-level adjustments through Event Browser. For information on using Event Browser for event-level adjustments, see "Adjusting Events".

About Adjusting Multiple Accounts Simultaneously

In some situations, you may want to adjust multiple A/R accounts simultaneously. For example, if regular daily rates were mistakenly applied during a holiday discount period, you would most likely want to perform adjustments for all the accounts that used the service over the holiday.

This type of adjustment is called a bulk adjustment and runs as a batch job initiated by operations personnel from the command line. The batch file for the bulk adjustment defines the accounts, balance groups, and amounts involved in the adjustment. The file is in CSV format and is typically supplied by an external source program. Bulk adjustments use the BRM multithreaded application (MTA) framework to process the batch file and perform the adjustment defined in this file for each listed account. For information on the MTA framework, see "Creating BRM Client Applications by Using the MTA framework" in *BRM Developer's Guide*.

The input file also specifies whether the adjustment is taxable. BRM bases tax calculation on the full adjustment amount for each account. Depending on how adjustment taxation is configured, BRM may apply the adjustment tax reversal when the bulk adjustment runs or send the adjustment amount for deferred taxation during the next billing run. For information on configuring taxes for adjustments, see "Configuring Taxes for Adjustments, Disputes, and Settlements".

When BRM processes a bulk adjustment, it performs each adjustment as a separate transaction. Therefore, there is no need to roll back the successful adjustments if some adjustments fail. BRM reports any accounts that failed the bulk adjustment in a log file so that you can examine them and correct the problem. All the CSV records that failed to process due to wrong format or due to the server side errors are written to an error file. The default name of this file is **pin_mta_failed.csv**. The names and locations of these files can be defined in the **pin_apply_bulk_adjustment pin.conf** file.

Bulk adjustments are initiated by operations personnel at the command line by using the **pin_apply_bulk_adjustment** utility and a CSV file.

About CSV Files

To perform a bulk adjustment, you must generate a CSV file that lists each of the accounts that should receive the bulk adjustment and provides information about the adjustment. BRM uses the CSV file to create an input flist for each of the adjustments that are part of the bulk adjustment.

CSV is a standard file format that uses commas as field delimiters and line breaks as record delimiters. CSV files should contain no blank lines. CSV files used for bulk adjustments are typically generated by an external program, and the records must use the following format:

```
account_POID, adjustment_amount, balance_group_POID, tax_flag, tax_code, tax_
supplier_ID, resource_ID, end_time, reason_code_domain, reason_code, description
```

Note: A CSV record should not contain the line break implied above. Line breaks should occur at the end of each record.

Consider these guidelines:

- The *account_POID*, *adjustment_amount*, and *resource_ID* fields are mandatory. You can omit any of the other fields. If you omit any field in a record (including those at the end of the record), you must still include the associated comma so BRM can keep track of which field it's processing.
- If you omit the *balance_group_POID*, BRM adjusts the default balance group.
- The *tax_flag* field indicates whether the adjustment requires tax reversal. If the flag is set to **1**, the adjustment does not need tax reversal and the generated input flist contains the PIN_AR_WITHOUT_TAX flag. If the flag is set to **2**, the adjustment involves a tax reversal and the generated input flist contains the PIN_AR_WITH_TAX flag instead. If you omit this field, the adjustment occurs without a tax reversal.

Note: The tax flag is ignored in any records that specify a non-currency resource.

- The *resource_ID* is the numeric code for the resource you are adjusting; for example, **840** for US dollars. You can specify currency or non-currency resources.
- The format for the *end_time* field is *MM/DD/YYYY*. BRM uses midnight for the date you specify as the time stamp for the adjustment.
- If you include a *reason_code_domain*, you must also include a *reason_code*, and the reverse.
- If the description field includes any commas, you must enclose the field in quotation marks ("). Quotation marks are optional for description fields that do not include commas.

The following CSV file segment shows a bulk adjustment with tax:

```
0.0.0.1 /account 15269 0, -9.5, 0.0.0.1 /balance_group 12901 0, 2, , , 840, , , , Rate issue
0.0.0.1 /account 12581 0, -9.5, 0.0.0.1 /balance_group 16165 0, 1, , , 840, , , , Rate issue
0.0.0.1 /account 15557 0, 10, , , , , 1000010, 04/26/2004, 12, 5, "Service drop, fix this"
```

In the above example, the first record specifies an adjustment with a tax reversal against the currency resource for a specific balance group. The adjustment amount is $9.50. The second record specifies a similar adjustment, except that this adjustment has no tax reversal. The last record specifies an adjustment of 10 free domestic minutes to the account's default balance group. BRM does not apply a tax reversal for this adjustment.

Running a Bulk Adjustment

To run bulk adjustments, generate a CSV file and then run the **pin_apply_bulk_adjustment** utility to load the file into the database.

Important: To connect to the BRM database, **pin_apply_bulk_adjustment** needs a configuration file in the *BRM_Home*/**apps/pin_bulk_adjust** directory. Or, if you are not starting the utility from this directory, you need the configuration file in directory from which you run the utility. For more information, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

---

1. Generate the CSV file and check it for format problems and spurious blank lines.
2. Use the following command to run the **pin_apply_bulk_adjustment** utility:

   ```
   pin_apply_bulk_adjustment -f input_file.csv
   ```

   Important: If the CSV file is not in your working directory, use the full path to the file:

   ```
   pin_apply_bulk_adjustment -f /files/bulk_adjust/input_file.csv
   ```

3. Check the pin_bulk_adjust.pin log file for any failed adjustments.

   The bulk adjustment application generates an intermediate file in flist format. The default name of this file is **pin_mta_search.flist**.

   All the CSV records that failed to process due to wrong format or due to the server side errors are written to an error file. The default name of this file is **pin_mta_failed.csv**.

   The names and locations of these files can be defined in the **pin_apply_bulk_adjustment pin.conf** file.

Adjustments and Corrective Invoicing

Any account-level adjustments or service-level adjustments for a past bill must be allocated to the prior bill *before* BRM generates the corrective bill for that bill. To ensure that the totals and balances for the bill item and the bill correctly reflect the corrections, enter all adjustments either manually or automatically through the rerating process before you generate the creative invoice for such a bill.

When it generates the corrective bill for such a prior bill, BRM includes all the A/R actions applied or allocated to that bill. It does not enable you to select some allocated A/R items and exclude others to be included in the next bill or the bill in progress.

Business Parameter for Allocating Automatic Adjustments from Rerating

The **AllocateReratingAdjustments** business parameter is used to specify whether BRM should allocate automatic adjustments from rerating to original bills.

Note: To ensure that the corrective billing process includes or excludes these adjustments, check the setting for this business parameter *before* you run the rerating process.

You can set **AllocateReratingAdjustments** to the following values:

- **disabled**: This is the default value. BRM does not make any automatic allocation during the rerating. It creates only unallocated adjustments with no information about the original items.
- **enabled**: BRM automatically allocates adjustments to the items on the original bill.

  For information on configuring **AllocateReratingAdjustments**, see *BRM Setting Up Pricing and Rating*.

Allocating Adjustments for Regular Billing

If you enable the **AllocateReratingAdjustments** business parameter, BRM displays the adjustment details by original item by original bill on the next bill.

Allocating Adjustments for Corrective Billing

When you enable **AllocateReratingAdjustments** business parameter, BRM allocates adjustments to the corrected bill items in the following manner:

- For open item accounting, the adjustment items are allocated to each bill that was corrected. And allocation is made to each of the original bills which included events or items that were corrected by these adjustments.
- For balance forward accounting, corrections are posted to the last bill only. Therefore, the rerating allocation is made to the final bill only. This final bill carries over the balances for all prior bill periods.

Adjustments and Corrective Invoices for Subordinate Hierarchies

The value of the **subARItemsIncluded** business parameter determines how BRM displays the adjustments in the summary and detailed versions of a **Replacement Invoice** or an **Invoice Correction Letter**.

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

About Disputes and Settlements

The dispute process involves two distinct activities; opening a dispute and settling that dispute. A dispute is a transaction that records a customer's objection to a currency amount billed to an account. A settlement is a transaction that resolves a dispute by crediting or debiting all, part, or none of the dispute amount to the account. Customers can also dispute events that use non-currency resources in the account. In this case, settling the dispute affects non-currency balances.

CSRs perform disputes and settlements at various levels, as appropriate to the situation. The way that BRM processes disputes and settlements varies slightly from level to level, as follows:

- **Bill level**: CSRs can dispute or settle an entire bill or a selection of bill items. In either case, BRM creates a single dispute or settlement item and alters the Due of each bill item covered by the dispute or settlement. Bill-level disputes and settlements can be thought of as a set of item-level disputes or settlements covered under the umbrella of a single dispute or settlement item.

  Bill-level disputes and settlements can only act against A/R bills. CSRs cannot dispute or settle a bill from a subordinate bill unit by filing a bill-level dispute or settlement directly against that bill. Instead, they file the bill-level dispute or settlement against the parent A/R bill. The dispute amount cannot exceed the total amount of the bill against which the dispute is applied, and the settlement amount cannot exceed the total dispute amount for the bill.

- **Item level**: Disputes and settlements at this level operate as either a CSR-initiated action against a single bill item or as a unilateral action against a set of bill items initiated by a bill-level dispute or settlement. Depending on how the dispute or settlement was initiated, BRM creates dispute and settlement items as follows:
  - If the CSR initiated the dispute or settlement at the item level, one dispute or settlement item is created for the item he or she chooses.
  - If the dispute or settlement was initiated at the bill level, a single dispute or settlement item will cover all the individual item-level disputes that make up the bill-level dispute.
- **Event level**: CSRs can dispute and settle any event and they can dispute or settle multiple events from an account in one operation.

  When an event-level dispute is opened, BRM creates one dispute event for each disputed event, establishing a one-to-one correspondence between the dispute event and the original event. The dispute event updates the original item’s Dispute field. BRM bundles all the individual dispute events into one dispute item.

  Similarly, when you settle an event-level dispute, BRM creates a settlement event for each dispute event. In this case, BRM transfers the settlement amount to the Adjusted field of the original item and the denied amount to the Due field. As with disputes, BRM bundles all individual settlement events into one settlement item.

BRM performs disputes and settlements at each of these levels by calling different dispute and settlement opcodes. For information on how to customize disputes and settlements and on which opcodes to call from your client application, see "Performing Adjustments with Your Custom Application".

CSRs open and resolve disputes for items and events by using Customer Center. For information about managing item-level and event-level disputes and settlements in Customer Center, see the Customer Center Help.

Note: CSRs cannot perform bill-level disputes and settlements through Customer Center. To dispute and settle bills, CSRs dispute and settle individual bill items, one by one. However, if you use a third-party CRM, you can implement disputes and settlements at the bill level by calling the appropriate opcode.

BRM system does not differentiate between bill-level and item-level disputes. Therefore, the person settling the dispute must keep track of the dispute type and settle the dispute with the corresponding settlement opcode.

Corrective Billing for Disputes, Settlements and Write-offs

Disputes, settlements, and write-offs impact the content of a corrective bill. If you must prevent the generation of corrective bills for certain A/R actions, set up a custom validation policy for BRM to use when you generate the corrective bills. Such a validation could be used to prevent BRM from generating corrective bills when the prior bill is in dispute and a settlement has not been reached.

Disputes

Corrective bills can be generated for a dispute entered at the bill level, item level or event level. When BRM generates the corrective bill for a prior bill in dispute, the corrective bill contains the details of the dispute, the disputed bill item under the corrected items, and the bill balance reduced by the disputed amount.

Settlements

Corrective bills can be generated for a dispute settled at the bill level, item level or event level. When BRM generates the corrective bill when a bill item on the prior bill is in settlement, the corrective bill contains the details of the settlement, the settled bill item under the corrected items, and the bill balance reduced or increased by the settlement.

A dispute may be settled for the full amount of the dispute in the customer’s favor. If you submitted a corrective bill at the time of such a dispute, the corrective bill at settlement time may become a duplicate as there may not be any additional corrections to the second bill.

Write-Offs

You perform write-offs after the account is past due and considered delinquent in collections. When BRM generates the corrective bill for a write-off on a bill item on the prior bill, the corrective bill contains the details of the write-off, the written-off bill item under the corrected items, and the bill balance reduced by the write-off amount.

Configuring Taxes for Adjustments, Disputes, and Settlements

You can specify whether to perform an adjustment, dispute, or settlement with or without taxes at the bill, item, and event level. For taxable account-level, bill-level and item-level adjustments, BRM uses information supplied by a tax configuration file to calculate the tax reversal for the account.

If you include taxes in an A/R action and the billing has already occurred, they are always calculated at the time the adjustment, dispute, or settlement is created. If the billing has not occurred, the calculated amount is applied only if the item has real-time taxes. If the item has deferred taxes, only the adjustment, dispute, or settlement amount is applied; the tax amount is calculated and applied at the time of billing.

Important: To ensure tax calculations are performed correctly for adjustments, disputes, settlements, and write-offs:

- Configure a separate item for taxes exclusively. To configure a separate item for taxes, see "Cumulative Custom Item for Taxes" in *BRM Configuring and Running Billing*.
- All events associated with an item must use the same tax method; tax now (real-time taxes) or tax deferred. Nontaxable events are valid in either taxation scenario.

BRM calculates tax as a percentage of the net amount (amount without taxes) of the original item. For example, consider a $110 item that has a 10% tax. The net amount is $100 and tax amount is $10. You perform a 50% adjustment with tax. In this case, the net adjustment amount is $50 and the tax reversal amount is $5 (10% of 50).

When BRM computes tax, it compares the calculated tax amount with the cycle tax’s due amount:

- If the calculated tax amount is greater than the cycle tax due amount, the cycle tax due amount is transferred to the tax item.
- If the calculated tax amount is less than or equal to the cycle tax due amount, the calculated tax amount is used.

Note: For untaxed adjustments, BRM adjusts the amount but does not calculate or apply a tax reversal for the adjustment amount.

To calculate adjustment, dispute, or settlement tax, BRM acquires tax information from one of two sources:

- For taxable adjustments, disputes, and settlements at the account, bill, and item level, BRM uses information supplied by a tax configuration file to calculate the tax reversal.

  Note: This tax configuration file supports only a single tax code, that is, for account-, bill- and item-level adjustments, the tax reversal would happen as per the single tax code mentioned in the configuration object.

- For taxable adjustments, disputes, and settlements at the event level, BRM obtains tax information from the event itself if BRM is configured for the tax now method. Otherwise, BRM obtains the taxes from the **/event/billing/cycle/tax** object.

In either case, this information typically includes the event type, tax supplier, and tax code. BRM uses this information in concert with the tax locale determined at bill run time to calculate the adjustment tax. For information on tax suppliers and tax codes, see "About Calculating Taxes" in *BRM Calculating Taxes*.

For BRM to correctly apply adjustment, dispute, and settlement taxes, you must configure tax information for events, define characteristics such as whether BRM should defer taxes, and set up tax defaults for Customer Center. For more information, see "About Configuring Taxes for Adjustments, Disputes, and Settlements".

About Adjustment Tax and Billing Cycles

If the adjustment occurs at the account, subscription service, or member service level, BRM uses the total amount of the adjustment, without consideration of the specific events that are included in the adjustment.

For bill-level, item-level, and event-level adjustments, BRM distinguishes between items and events that need real-time taxes and those that need deferred taxes. If some adjusted items or events are nontaxable, BRM omits these from the deferred amount that it will transfer for taxation during billing. For partial adjustments, BRM also proportions the deferred amount according to how much of an adjustment was actually granted.

About Dispute and Settlement Taxes and Billing Cycles

For bill-level and item-level disputes and settlements, real-time taxes are applied at the time the dispute or settlement is created. When processing a dispute or settlement with deferred taxes, BRM determines whether the original item has been billed. If billing has not yet occurred, BRM has not levied the tax for the original item. In this case, there is no need apply the taxes separately because the dispute or settlement will have acted on the balance of the original item before BRM ever calculates the taxes for that item.

However, if the dispute or settlement is opened against an item that has already been billed, the customer has already been charged tax for the item. In this case, BRM must specifically accommodate dispute or settlement tax. BRM defers applying the tax reversal amount until billing is run.

For partial disputes and settlements, BRM proportions the deferred amount in accordance with the relative size of the dispute or settlement.

About Configuring Taxes for Adjustments, Disputes, and Settlements

Depending on tax law, jurisdictions, and other factors, you may need to specify whether to apply a tax reversal when performing certain adjustments, disputes, and settlements. For example, the law may require that you reverse taxes for any adjustment you apply to a charge for a call placed outside the country, but not require you to reverse taxes for adjustments to prepaid balances.

As a prerequisite for all but the event-level adjustment, dispute, and settlement opcodes, you must load a **/config/ar_taxes** object that defines tax treatment parameters for each event type. BRM uses this object to enrich the adjustment, dispute, and settlement opcode flists with the tax information needed to calculate the tax reversal. You load **/config/ar_taxes** by running a configuration utility against a **pin_config_ar_taxes.xml** file you create. For more information, see "About the pin_config_ar_taxes.xml File".

To specify whether tax reversals occur when performing certain adjustments, disputes, and settlements, set the value of the PIN_FLD_FLAGS field in the input flist of the appropriate opcode. See "Configuring the Tax Treatment for Adjustments, Disputes, and Settlements".

To specify that tax reversals occur at the time account-level adjustments are created rather than deferring them until billing, modify the Connection Manager (CM) **pin.conf** file. For event, item, and bill level A/R actions, the time at which tax reversals occur is determined by the rate plan associated with the event. See "Configuring the Default Tax Method for Account-Level Adjustments".

If you use Customer Center, the tax reversal entry in the **pin.conf** file is ignored. Instead, you use the Customer Center Configurator to set up the tax treatment. See "Configuring the Default Tax Treatment for Customer Center".

About the pin_config_ar_taxes.xml File

The **pin_config_ar_taxes.xml** file provides information on the tax supplier and tax code for each adjustment, dispute, and settlement event type.

- The tax suppliers you specify in the **pin_config_ar_taxes.xml** file must be listed in the **tax_supplier_map** file.
- The tax code you use in the **pin_config_ar_taxes.xml** file must be present in the **taxcodes_map** file. The tax code is mapped to a package that calculates tax for that product type, and can be a numeric or descriptive string.

For information on tax suppliers, tax codes, the associated map files, see "About Supplying the Data Needed for Calculating Taxes" in *BRM Calculating Taxes*.

The **pin_config_ar_taxes.xml** file must follow all standard XML formatting rules. The following example shows a typical **pin_config_ar_taxes.xml** file.

Note: The <Name> field in the sample identifies the tax supplier.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <AccountReceivablesConfiguration>
    <TaxConfigurationList>
      <TaxConfiguration>
        <Event>/event/billing/adjustment/item</Event>
        <Name>ABC Inc - California branch</Name>
        <TaxCode>usage</TaxCode>
      </TaxConfiguration>
      <TaxConfiguration>
        <Event>/event/billing/dispute/item</Event>
        <Name>ABC Inc - California branch</Name>
        <TaxCode>usage</TaxCode>
      </TaxConfiguration>
      <TaxConfiguration>
        <Event>/event/billing/settlement/item</Event>
        <Name>ABC Inc - New Jersey branch</Name>
        <TaxCode>usage</TaxCode>
      </TaxConfiguration>
      <TaxConfiguration>
        <Event>/event/billing/adjustment/account</Event>
        <Name>ABC Inc - Illinois branch</Name>
        <TaxCode>cycle</TaxCode>
      </TaxConfiguration>
    </TaxConfigurationList>
  </AccountReceivablesConfiguration>
```

Loading the Tax Configuration into the BRM Database

Important: Before you load the tax configuration, you must first load the tax suppliers. See "About Defining Tax Suppliers" in *BRM Calculating Taxes*.

To load the tax configuration, edit the sample **pin_config_ar_taxes.xml** file, then run the "load_pin_ar_taxes" utility to load the contents into the **/config/ar_taxes** object in the BRM database:

Important: To connect to the BRM database, **load_pin_ar_taxes** needs a configuration file in the directory from which you run the utility. For more information, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Caution: The **load_pin_ar_taxes** utility overwrites the existing tax information that BRM uses for the various adjustment and dispute events types. If you are updating the current tax information base to add event types or change the tax specifications for an event type, you cannot load the new tax information only. You must load complete sets of tax information for all event types each time you run the **load_pin_ar_taxes** utility.

1. Edit the sample **pin_config_ar_taxes.xml** file in the *BRM_Home***/sys/data/config/** directory.

   Note: You can rename the file and save it at a location of your choice.

2. Save the **pin_config_ar_taxes.xml** file.
3. Use the following command to run the **load_pin_ar_taxes** utility:

   ```
   load_pin_ar_taxes input_file.xml
   ```

   If the tax configuration XML file is not in your working directory, use the full path to the file:

   ```
   load_pin_ar_taxes -f /files/tax_config/input_file.xml
   ```

4. Stop and restart the Connection Manager (CM) and, if necessary, your client application.

   For more information on restarting CM, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the **pin_config_ar_taxes.xml** file was loaded, you can display the **/config/ar_taxes** object by using the Object Browser, or use the **robj** command with the **testnap** utility. (See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.)

Configuring the Default Tax Method for Account-Level Adjustments

By default, BRM defers calculating tax for all account-level adjustments until the end of the billing cycle. To change this so that BRM calculates tax when the adjustment is created, add an entry to the **pin.conf** file.

To enable the tax now feature for account-level adjustments:

1. Open the Connection Manager configuration file (*BRM_Home***/sys/cm/pin.conf)**.
2. To specify the tax now method, add the following line to the file:

   ```
   -fm_ar tax_now 1
   ```

   Note: If this line is not present in the file, is commented out, or is set to **0**, BRM uses the deferred tax method for all account-level adjustments.

   Important: To ensure tax calculations are performed correctly, all events associated with an item must use the same tax method; tax now or tax deferred. Tax-exempt events are valid in either taxation scenario.

You do not need to restart the CM to enable this entry.

Configuring the Tax Treatment for Adjustments, Disputes, and Settlements

To ensure that you can control whether BRM applies adjustment, dispute, and settlement tax reversals, your CRM implementation should set the following flags in the input flists for all account, bill, and item level adjustment, dispute, and settlement opcodes:

- **PIN_AR_WITH_TAX**: Set this flag for all adjustments, disputes, and settlements that involve a tax reversal.
- **PIN_AR_WITHOUT_TAX**: Set this flag for adjustments, disputes, and settlements that do not require a tax reversal.

If you do not pass in one of the flags, BRM uses the **fm_ar tax_reversal_with_tax** entry in the CM configuration file to determine the default behavior. This entry takes one of the tax flags (PIN_AR_WITH_TAX or PIN_AR_WITHOUT_TAX). If you do not pass in one of the flags and this entry is not present, BRM does not perform a tax reversal.

Note: Event-level adjustments, disputes, and settlements use different flags for tax reversal. For more information on tax flags for event-level opcodes, see "Flags for Event Adjustments", "Flags for Event Disputes", and "Flags for Event Settlements".

Configuring the Default Tax Treatment for Customer Center

If you are using Customer Center as your CRM application, you can configure a default tax treatment for adjustments, disputes, and settlements. Customer Center uses this configuration to determine whether these activities always have a tax reversal, never have a tax reversal, or have a tax reversal at the CSRs discretion. Configuring a default tax treatment can help you enforce uniform tax treatments for a given location and eliminate the need for a decision by the CSR.

You configure the tax treatment on the **Balance** tab in the Customer Center Configurator. This configuration overrides any tax treatment specification in the **pin.conf** file. To set up tax treatment for adjustments, disputes, and settlements, select one of the following options under **Tax treatment** on the Configurator **Balance** tab:

- **Include tax**: Always perform a tax reversal for the adjustment, dispute, or settlement.
- **Exclude tax**: Never perform a tax reversal for the adjustment, dispute, or settlement.
- **None**: Allow the CSR to choose whether to include taxes.

For information on using the Customer Center Configurator **Balance** tab and the implications of these three options, see "Balance Configurator" in *BRM Developer's Guide*.

Reserving and Freeing Resources for Disputes and Settlements

To ensure that resources are not misused when a dispute is open, BRM reserves resources disputed at the event level. The reservation process protects the account balance from being credited for the dispute amount until settlement occurs. Resource reservation is particularly important for prepaid accounts, where the dispute amount might incorrectly be added to prepaid resources. For example, if the customer disputed a 10 minute charge on his or her bill, this amount would be credited to the account balance and, for prepaid accounts, would increase the prepaid amount. However, placing a reservation on that 10 minutes lowers the credit limit for the account and prevents the disputed time from being freely used as it would be in a normal credit situation.

When you open a dispute, BRM uses Resource Reservation Manager and new policies you define to create a reservation object for each dispute event. The reservation object prevents the resource associated with the event from being credited to the account; in essence, locking the resource amount until settlement. Upon settlement of the dispute, BRM releases each reservation associated with the dispute and credits the account by subtracting the settlement amount from the Due amount.

BRM performs event-level resource reservation for disputes and settlements using these two policy opcodes:

- PCM_OP_RESERVE_POL_POST_DISPUTE creates a **/reservation** object for each entry in the balance impact arrays for each of the events associated with the dispute item.
- PCM_OP_RESERVE_POL_POST_SETTLEMENT changes the PIN_FLD_ RESERVATION_STATUS field in each of the **/reservation** objects for the dispute from 0 to 1, indicating that the resource is now available.

These two opcodes are called by PCM_OP_ACT_USAGE whenever it processes an event-level dispute or settlement, provided the following requirements are met:

- You have Resource Reservation Manager, an optional BRM component.
- Event notification is enabled in your system.

  See "Configuring Event Notification for Disputes and Settlements".

Configuring Event Notification for Disputes and Settlements

When an event-level dispute is opened or settled, BRM uses event notification to call opcodes that perform the appropriate follow-up operations.

Although any subclass of the **/event** class can be used to trigger event notification, the dispute and settlement opcodes generate the following events specifically to use for event notification:

- **/event/billing/dispute/notify**: By default, when PCM_OP_AR_EVENT_DISPUTE generates this event and passes it to PCM_OP_ACT_USAGE, PCM_OP_ RESERVE_POL_POST_DISPUTE is called.
- **/event/billing/settlement/notify**: By default, when PCM_OP_AR_EVENT_ SETTLEMENT generates this event and passes it to PCM_OP_ACT_USAGE, PCM_OP_RESERVE_POL_POST_SETTLEMENT is called.

Before you can use the event-level dispute and settlement feature, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them.

   See "Merging Event Notification Lists" in *BRM Developer's Guide*.

2. Ensure that the merged file includes the following information from the *BRM_ Home*/**sys/data/config/pin_notify** file:

   ```
   # Settlements and disputes related event notifications
   2354    0    /event/billing/settlement/notify
   2355    0    /event/billing/dispute/notify
   ```

3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.

   See "Editing the Event Notification List" in *BRM Developer's Guide*.

4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger.

   See "Triggering Custom Operations" in *BRM Developer's Guide*.

5. Load your final event notification list into the BRM database.

   See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Working with Reason Codes for Adjustment, Disputes, and Settlements

Reason codes explain why an adjustment, dispute, or settlement is being granted. In typical implementations, the BRM client application provides the CSR with a selectable set of reasons appropriate to the event type. The CSR chooses a reason that matches the situation or, in some cases, composes a new reason. BRM populates the input flist for the opcode with this information, and the reason is recorded in the **/event** object that the opcode creates for the adjustment, dispute, or settlement.

Reason codes are stored in the objects based on the **/event/billing** class, which is the parent class for all adjustment, dispute, and settlement event objects (for example, **/event/billing/adjustment/item**, **/event/billing/dispute/event**, and so forth). You typically select from a list of reasons appropriate to the event type, as defined in the reasons.locale configuration file. When Connection Manager starts, BRM uses this file to construct the list of valid adjustment, dispute, or settlement reasons for each event type.

For event-level adjustments, BRM determines whether the reason code you select makes sense for each event in a batch. If the reason for the adjustment is not appropriate for a given event, that event is omitted from the adjustment. For example, if the batch of events to be adjusted includes both dial-up and telco events, and the reason supplied for the adjustment is that invalid roaming charges were applied, BRM excludes the dial-up events from the adjustment since a dial-up service is never subject to roaming charges.

To implement reason codes, you create and load a **reasons**.*locale* file that lists each reason code and associates it with a reason code domain. The domain classifies the reason code according to the type of activity it applies to. You can also include information in the **reasons**.*locale* file that prevents inappropriate event-level adjustments based on reason code and service.

About the reasons.*locale* File

The **reasons**.*locale* file defines each reason code domain, the reason codes that belong to the domain, and the event G/L ID. The domain and reason code information is used to build the **/strings** object and the event G/L ID is used to build the **/config/map_glid** object.

For event-level adjustments, you can also define which services are valid for the reason code and specify the valid event types within those services. This information is used to build the **/config/reason_code_scope** object. For each event in an event-level adjustment batch, BRM checks **/config/reason_code_scope** to determine whether the associated service and event type are included in the PIN_FLD EVENT array. If they are not, BRM skips the adjustment for that event.

The following example shows a single **reasons**.*locale* file segment defining a reason code domain.

Note: The service block is optional; it is used to define the reason code scope. If you do not include it, BRM adjusts all events in the batch regardless of whether the reason for the adjustment makes sense for the service and event.

```
DOMAIN = "Reason Codes - Event Adjustment Reasons";
STR
  ID = 3;
  VERSION = 22;
  STRING = "Net Speed Was Slow";
  SERVICE
    TYPE = "/service/ip";
    EVENT
      TYPE = "/event/session";
    EVENT-END
    EVENT
      TYPE = "/event/session/dialup";
    EVENT-END
  SERVICE-END
  SERVICE
    TYPE = "/service/telco/gsm/data";
    EVENT
      TYPE = "/event/session/gsm/telco";
    EVENT-END
  SERVICE-END
  EVENT-GLID
    "/event/billing/adjustment/event"         105;
  EVENT-GLID END
END
```

For more information on the **reasons**.*locale* file, see "String Manipulation Functions" in *BRM Developer's Reference*.

Loading Adjustment, Dispute, and Settlement Reason Codes into the BRM Database

To define reason codes for adjustments, disputes and settlements, you edit the **reasons.en_US** sample file in the *BRM_Home***/sys/msgs/reasoncodes** directory. You then use the **load_localized_strings** utility (see "load_localized_strings" in *BRM Developer's Guide*) to load the contents of the file into the **/strings**, **/config/map_glid**, and **/config/reason_code_scope** objects.

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

```
load_localized_strings reasons.locale
```

Note:

- If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.
- If you add your own reason codes to the **reasons**.*locale* file, you should use IDs above 100,000.

Caution: The **load_localized_strings** utility overwrites existing **/config/map_glid** and **/config/reason_code_scope** objects. If you are updating these objects, you cannot load new G/L ID maps and reason code scopes only. You must load complete sets of data each time you run the **load_localized_strings** utility. This is also true when using the **/strings** object, but only if you specify the **-f** parameter. Otherwise, the **load_localized_strings** utility appends the new data to the object.

For information on loading the **reasons.***locale* file, see "Loading Localized or Customized Strings" in *BRM Developer's Guide*. For information on creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

Performing Adjustments with Your Custom Application

You can perform adjustments at a variety of levels by calling different adjustment opcodes. Calling the full range of these opcodes through your BRM client application gives you the flexibility of making anything from low-level adjustments against specific events to high-level adjustments distributed across an A/R account and its child accounts. For example, you might implement different adjustment levels to enable the CSR to help a customer who was charged too many minutes for a call and a customer whose prepaid service experienced a 3-day outage.

For background information about adjustments, see "About Adjustments".

For information about performing adjustments in Customer Center, see the Customer Center Help.

Use the following opcodes to perform and customize adjustments:

- PCM_OP_AR_ACCOUNT_ADJUSTMENT

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

  See "Adjusting Bills".
- PCM_OP_AR_ITEM_ADJUSTMENT

  See "Adjusting Items".
- PCM_OP_BILL_POL_VALID_ADJUSTMENT

  See "Customizing Item-Level Adjustments".
- PCM_OP_AR_EVENT_ADJUSTMENT

  See "Adjusting Events".
- PCM_OP_ACT_POL_SPEC_GLID

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

Adjusting Accounts, Subscription Services, and Member Services

To make adjustments at the account level, subscription service level, or member service level, use PCM_OP_AR_ACCOUNT_ADJUSTMENT. This opcode debits or credits the currency resource balance for the specified account.

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

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

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

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

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

If an account, subscription service, or member service adjustment is for non-currency resources, use the PCM_OP_BILL_DEBIT opcode to adjust the non-currency resource. For more information on how PCM_OP_BILL_DEBIT works with resources, see "Applying Debits and Credits".

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

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

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

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

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

- PIN_FLD_FLAGS for the tax treatment:
  - PIN_AR_WITHOUT_TAX: The account adjustment is nontaxable.
  - If the PIN_AR_WITH_TAX: The account adjustment is taxable.

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

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

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_ACCOUNT_ADJUSTMENT.

Flags for Account, Subscription Service, and Member Service Adjustments

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

Adjusting Bills

To make adjustments at the bill level, use PCM_OP_AR_BILL_ADJUSTMENT. This opcode debits or credits a currency balance for the specified **/bill** object. You can open a bill-level adjustment with or without tax. For information on how PCM_OP_AR_BILL_ADJUSTMENT performs adjustment taxation, see "Including Taxes in the Adjustment".

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

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

PCM_OP_AR_BILL_ADJUSTMENT *does not* make adjustments when:

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

However, if a bill is fully or partially paid, PCM_OP_AR_BILL_ADJUSTMENT *does not* make adjustments when the amount of the adjustment exceeds the remaining due amount of the bill. To use PCM_OP_AR_BILL_ADJUSTMENT to adjust the total amount of a fully or partially paid bill, see "Adjusting Fully or Partially Paid Bills".

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

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

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

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

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

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

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

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

Fields You Should Include in the Flist for Bill Adjustments

When performing bill-level adjustments, set these fields in the PCM_OP_AR_BILL_ADJUSTMENT input flist:

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

  Important: This field must be at level 1 of the input flist in order for the item adjustment to work.

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

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

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

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

  Note: If the bill adjustment is taxable, the absolute value of PIN_FLD_AMOUNT must be less than or equal to the total bill amount excluding the tax amount.

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

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_BILL_ADJUSTMENT.

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

Flags for Bill Adjustments

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

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

Adjusting Fully or Partially Paid Bills

You can use PCM_OP_AR_BILL_ADJUSTMENT to perform bill-level adjustments for fully or partially paid bills. By default, PCM_OP_AR_BILL_ADJUSTMENT makes adjustments only up to the remaining due amount of a bill if the bill is fully or partially paid.

To use PCM_OP_AR_BILL_ADJUSTMENT to make adjustments up to the total amount of a fully or partially paid bill, set the **BillPaymentDeallocation** field in the **ar** instance of the **/config/business_params** object to **enabled**. See "Improving Bill-Level Adjustments for Fully or Partially Paid Bills".

When the **BillPaymentDeallocation** field is enabled, you can use PCM_OP_AR_BILL_ADJUSTMENT to adjust the total bill amount for a fully or partially paid bill. The amount of adjustment for the bill can be more than the remaining due, but it cannot exceed the total bill amount. For example, if the total bill amount is $5 and you have already made a partial payment of $2, the remaining due is $3. In this scenario, you can perform a credit adjustment for up to $5 (total bill amount).

However, if there is an A/R action (such as an adjustment or a dispute) already performed on the bill, you can perform adjustments for only up to the amount due after the initial A/R action is performed. For example, if an item-level adjustment of $1 is made on a bill of $5 and you have already made a partial payment of $2, the remaining due is $2. In this scenario, you can perform a credit adjustment for only up to $4 (which is the amount due after item-level adjustment).

In case if the payment already made for the bill is more than the bill amount due after adjustment, the excess amount is left unallocated and it is displayed as **Adjustment/payments not applied** in Customer Center.

Improving Bill-Level Adjustments for Fully or Partially Paid Bills

By default, PCM_OP_AR_BILL_ADJUSTMENT does not perform bill-level adjustments for an amount more than the remaining due amount if the bill is fully or partially paid.

You can improve bill-level adjustments for fully or partially paid bills by setting the **BillPaymentDeallocation** field in the **ar** instance of the **/config/business_params** object to **enabled**.

To improve bill-level adjustments for fully or partially paid bills:

1. Use the following command to create an editable XML file for the BusParamsAR parameter class:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

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

2. Search the XML file for following line:

   ```
   <BillPaymentDeallocation>disabled</BillPaymentDeallocation>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **/config/business_params** object for the **ar** instance. If you delete or modify any other parameters in the file, these changes affect the associated aspects of BRM's billing configuration.

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

   ```
   pin_bus_params bus_params_AR.xml
   ```

   You should execute this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see **pin_bus_params**.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

6. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in the *BRM System Administrator's Guide*.

Adjusting Items

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

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

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

PCM_OP_AR_ITEM_ADJUSTMENT does the following:

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

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

2. Calls the PCM_OP_BILL_POL_VALID_ADJUSTMENT policy opcode to validate the changes.

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

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

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

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

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

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

4. If the adjustment has tax implications and the opcode is not being called as part of a bill-level adjustment, checks the original **/item** object to determine the taxable portion of the item:

   - If the item object stores only an amount generated by a tax event, the opcode omits it from tax calculation.

   - If the item object stores only an amount generated by a usage event, the opcode calculates taxes for the amount.

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

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

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

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

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

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

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

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

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

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

   If an over-adjustment is made on an item, the following occurs:

   - For deferred taxation, the adjustment amount is left unallocated in the adjustment item. If the tax to be adjusted is greater than the cycle tax amount, then the cycle tax item is adjusted for its due amount.
   - For tax now taxation, the excess adjustment amount is left unallocated in the adjustment item. This appears in Customer Center as **Adjustment/payments not applied**.

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

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

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

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

Fields You Should Include in the Input Flist for Item Adjustments

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

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

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

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_ITEM_ADJUSTMENT.

Flags for Item Adjustments

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

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

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

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

Customizing Item-Level Adjustments

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

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

Adjusting Events

To make adjustments at the event level, use PCM_OP_AR_EVENT_ADJUSTMENT. This opcode debits or credits a resources for a particular event.

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

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

By default, the adjustable events are:

- **/event/billing/debit**
- **/event/billing/product/fee/cycle/cycle_arrear**
- **/event/session/dialup**

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

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

When PCM_OP_AR_EVENT_ADJUSTMENT receives a batch of events, it:

1. Identifies all events in the batch to be adjusted, and runs the validation checks against those events.
2. If the event has any tax implications, the tax implications must be adjusted.

   For information on how PCM_OP_AR_EVENT_ADJUSTMENT performs adjustment taxation, see "Including Taxes in the Adjustment".
3. Creates a single **/item/adjustment** object for all the events in the batch for the account.
4. For each event to be adjusted for an account, a new **/event/billing/adjustment/event** object containing the amount to be adjusted is created.

   The /**event/billing/adjustment/event** object has the PIN_FLD_ITEM_OBJ pointing to the adjustment item created in step 3.
5. After the adjustment item and the /**event/billing/adjustment/event** object are created, the amount is transferred from the adjustment item to the original item where the adjusted event belongs.

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

Note: Once the entire Due amount from the adjustment item is transferred to other items, the adjustment item will be closed.

BRM creates a G/L ID for event-level adjustments.

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

Fields You Should Include in the Input Flist for Event Adjustments

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

- PIN_FLD_POID for each of the events to be adjusted. The adjusted amount is applied to the default balance group of the bill unit associated with the **/item** object that the event belongs to.

- PIN_FLD_AMOUNT or PIN_FLD_PERCENT for the adjustment amount. The value supplied for this field should be positive for adjustments that credit the event and negative for adjustments that debit the event. If you omit both of these fields, BRM adjusts the entire amount for each event.
- PIN_FLD_RESOURCE_ID for the resource to be adjusted.

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

- For information on tax treatment, see "Including Taxes in the Adjustment".
- For information on reason codes, see "Including Reason Codes in the Adjustment".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_EVENT_ADJUSTMENT.

Flags for Event Adjustments

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

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

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

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

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

Categorizing Unbilled Event-Level Adjustments in G/L Reports

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

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

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

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

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

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

Including Taxes in the Adjustment

You can specify an optional tax treatment for adjustments through flags in the input flist. For information, see:

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

Tax Processing for Account-Level Adjustments

If the input flist flags the adjustment as taxable or the **pin.conf** file specifies that BRM tax all adjustments, the opcode obtains tax information from a configuration file read into BRM when Connection Manager starts. For information on loading this tax information into BRM, see "Loading the Tax Configuration into the BRM Database".

The PCM_OP_AR_ACCOUNT_ADJUSTMENT opcode uses this information as follows:

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

Tax Processing for Bill and Item Level Adjustments

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

The PCM_OP_AR_ITEM_ADJUSTMENT and PCM_OP_AR_BILL_ADJUSTMENT opcodes perform either of the following two actions:

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

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

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

Tax Processing for Event-Level Adjustments

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

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

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

Including Reason Codes in the Adjustment

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

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

Reason code validation produces these results:

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

For general information about reason codes, see "String Manipulation Functions" in *BRM Developer's Reference*. For information about setting up reason codes for adjustments, see "Working with Reason Codes for Adjustment, Disputes, and Settlements".

Assigning G/L IDs for an Adjustment

The G/L ID for an adjustment event is assigned like other pre-rated events in the BRM system. The PCM_OP_ACT_POL_SPEC_GLID policy opcode assigns the G/L ID for the adjustment. The PIN_FLD_GL_ID field in the **/event/billing/adjustment/event** object points to the same G/L ID as the original event, ensuring that both the original event and the adjustment are correctly recorded in the general ledger. For information on preconfiguring the adjustment event to modify the **/config/map_glid** configuration object, see "Assigning G/L IDs to Nonrated Events" in *BRM Collecting General Ledger Data*.

Applying Debits and Credits

For background information, see "About Adjustments".

To apply debits and credits to the non-currency resources, use PCM_OP_BILL_DEBIT.

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

There are three ways to use PCM_OP_BILL_DEBIT:

- Pass in the POID of an account to impact that account's default **/balance_group** object.
- Pass in the POID of a specific balance group to impact that **/balance_group** object.

- For either the account or a balance group, pass in a specific sub-balance to impact that sub-balance.

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

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

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

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

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

Note:

- You can change the order in which PCM_OP_BILL_DEBIT updates sub-balances. For information, see "Specifying the Order in Which Resource Sub-Balances Are Consumed" in *BRM Setting Up Pricing and Rating*.
- PCM_OP_BILL_DEBIT opens a local transaction.

Flags for Applying Debits and Credits

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

Performing Disputes and Settlements with Your Custom Application

You can create disputes and settlements at a variety of levels by calling different dispute and settlement opcodes. As with adjustments, using the full range of these opcodes lets you process anything from a low-level dispute for a specific event to a higher-level dispute against an entire bill.

For information on creating disputes and settlements in Customer Center, see the Customer Center Help.

Use the following opcodes to perform and customize disputes and settlements:

- PCM_OP_AR_BILL_DISPUTE

  See "Disputing Bills".
- PCM_OP_AR_BILL_SETTLEMENT

  See "Settling Disputed Bills".
- PCM_OP_AR_ITEM_DISPUTE

  See "Disputing Items".
- PCM_OP_AR_ITEM_SETTLEMENT

  See "Settling Disputed Items".
- PCM_OP_BILL_POL_VALID_DISPUTE

  See "Customizing Item-Level Disputes".
- PCM_OP_AR_EVENT_DISPUTE

  See "Disputing Events".
- PCM_OP_AR_EVENT_SETTLEMENT

  See "Settling Disputed Events".
- PCM_OP_ACT_POL_SPEC_GLID

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

Disputing Bills

To open a dispute at the bill level, use PCM_OP_AR_BILL_DISPUTE. This opcode debits or credits a currency balance for the specified **/bill** object. You can open a bill-level dispute with or without tax. For information on how PCM_OP_AR_BILL_DISPUTE performs dispute taxation, see "Including Taxes in the Dispute or Settlement".

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

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

PCM_OP_AR_BILL_DISPUTE *does not* create disputes when:

- The specified bill is not an A/R bill; for example, a bill from a nonpaying subordinate bill unit.

- The amount of the dispute exceeds the total amount of the bill against which the dispute applies. For example, if the bill due is $5, you cannot open a credit dispute for $6. Similarly, if the bill due -$5, you cannot open a debit dispute for $6.

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

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

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

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

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

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

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

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

Fields You Should Include in the Input Flist for Bill Disputes

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

- PIN_FLD_POID for the bill under dispute. You obtain this from the **/bill** object that the CSR identifies through the client application. The disputed amount is applied to the default balance group of the bill unit associated with the bill.

- PIN_FLD_POID for the item under dispute. This field is only required to dispute a particular set of bill items within the bill.
- PIN_FLD_AMOUNT for the dispute amount. The value supplied for this field should be negative when crediting and positive when debiting. This field is optional.

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

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

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_BILL_DISPUTE.

Flags for Bill Disputes

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

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

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

Settling Disputed Bills

To settle a dispute at the bill level, use PCM_OP_AR_BILL_SETTLEMENT. This opcode debits or credits a currency balance for the specified **/bill** object. You can create a bill-level settlement with or without tax. For information on how PCM_OP_AR_BILL_SETTLEMENT performs settlement taxation, see "Including Taxes in the Dispute or Settlement".

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

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

PCM_OP_AR_BILL_SETTLEMENT *does not* create settlements in these cases:

- The specified bill is not an A/R bill; for example, a bill from a nonpaying subordinate bill unit.
- The amount of the settlement exceeds the amount of the associated dispute.

Bill-level settlements can apply to the entire bill or to specific bill items. When PCM_OP_AR_BILL_SETTLEMENT receives an input flist, it creates a single, umbrella settlement item for all bill items covered by the settlement, as follows:

- If the input flist specifies individual bill items, the settlement item is associated with each of these bill items. The settlement amount for each bill item must be specified in the input flist.
- If the flist does not specify any bill items, all disputed items in the bill are eligible for settlement. PCM_OP_AR_BILL_SETTLEMENT opens settlements for as many disputed items as it can before it consumes the settlement amount. After it uses up the settlement amount, it continues to open settlements for the remaining disputed items, but with a settlement amount of 0.

For each bill item it must settle, PCM_OP_AR_BILL_SETTLEMENT passes the POID of the settlement item and the appropriate settlement amount to PCM_OP_AR_ITEM_SETTLEMENT, which creates a settlement event for each bill item it receives. An **/event/billing/settlement/item** object is created for each settled item in the bill, recording the balance impact. For details on how PCM_OP_AR_ITEM_SETTLEMENT works, see "Settling Disputed Items".

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

BRM creates a G/L ID for every bill-level settlement. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist

Set these fields in the PCM_OP_AR_BILL_SETTLEMENT input flist when creating bill-level settlements:

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

  You can provide the PIN_FLD_AMOUNT field for the bill itself or, if the settlement is against an array of bill items, for the individual bill items. If you supply a value for PIN_FLD_AMOUNT, it must be less than the dispute amount.

PCM_OP_AR_BILL_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so forth.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_BILL_SETTLEMENT.

Flags for Bill Settlements

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

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

For information on flags that indirectly affect PCM_OP_AR_BILL_SETTLEMENT, see "Flags for Item Settlements".

Disputing Items

To open a dispute at the item level, use PCM_OP_AR_ITEM_DISPUTE. This opcode debits or credits a currency resources for a particular bill item. This opcode is also called by PCM_OP_AR_BILL_DISPUTE to dispute items in a bill.

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

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

PCM_OP_AR_ITEM_DISPUTE does the following:

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

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

2. Calls PCM_OP_BILL_POL_VALID_DISPUTE to validate the changes.

   You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT. For information on customizing the opcode, see "Customizing Item-Level Disputes".

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

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

   - Whether the field contains a POID, indicating that the opcode is being called by PCM_OP_AR_BILL_DISPUTE to complete a bill-level dispute.

     This POID identifies the **/item/dispute** object that PCM_OP_AR_BILL_DISPUTE created to store information on the bill-level dispute. This is the **/item/dispute** object that PCM_OP_AR_ITEM_DISPUTE will associate with each of the dispute events it creates for the bill-level adjustment.

   - Whether the field is NULL, indicating that PCM_OP_AR_ITEM_DISPUTE is being called directly from the client application to perform an item-level dispute. In this case, PCM_OP_AR_ITEM_DISPUTE will create its own **/item/dispute** object. This object is exclusive to the individual item being disputed.

4. If the dispute has tax implications and the opcode is not being called as part of a bill-level dispute, checks the original **/item** object to determine the taxable portion of the item:

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

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

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

   Note: If the item dispute is being performed as part of a bill-level dispute, the PCM_OP_AR_BILL_DISPUTE opcode performs this step before calling PCM_OP_AR_ITEM_DISPUTE.

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

   The item dispute event balance impact is applied to the default balance group of the bill unit associated with the item. If the item dispute is part of a bill-level dispute, the event object for each disputed item is associated with a single, umbrella **/item/dispute** object.

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

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

7. Calls PCM_OP_BILL_TRANSFER to transfer funds to the A/R bill.
8. Writes the modified version of the original **/item** object.
9. Provides feedback on whether the dispute was successfully created.

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

BRM creates a G/L ID for item-level disputes. For information on disputes and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist

Set these fields in the PCM_OP_AR_ITEM_DISPUTE input flist when creating item-level disputes:

- PIN_FLD_POID for the item under dispute. The disputed amount is applied to the default balance group of the bill unit associated with the item.
- PIN_FLD_AMOUNT for the dispute amount. The value for this field should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the resource to be disputed.

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

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_ITEM_DISPUTE.

Flags for Item Disputes

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

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

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

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

Customizing Item-Level Disputes

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

You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

Settling Disputed Items

To settle a dispute at the item level, use PCM_OP_AR_ITEM_SETTLEMENT. This opcode debits or credits the currency balance of a particular item to settle a dispute against that item. This opcode is also called by PCM_OP_AR_BILL_SETTLEMENT to settle the dispute items associated with a bill.

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

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

PCM_OP_AR_ITEM_SETTLEMENT does the following:

1. Reads the original **/item** object and prepares the changes to reflect the balance impact based on the settlement amount.

   Settlement amounts have components:

   - **Granted amount**: The portion of the disputed amount granted by the settlement. This is the amount in the PIN_FLD_DISPUTED field of the PCM_OP_AR_ITEM_SETTLEMENT input flist.
   - **Denied amount**: The portion of the disputed amount that was denied by the settlement. This is the original dispute amount minus the granted amount. Settlements always work with the denied amount.

   For credit settlements, PCM_OP_AR_ITEM_SETTLEMENT makes the following changes to the balance impact:

   - Increases the Due amount (PIN_FLD_DUE) for the item by adding the amount denied during the settlement, if any.
   - For item-level disputes, it impacts the Adjusted amount (PIN_FLD_ADJUSTED) of the bill item by adding the settled amount and zeroing out the disputed amount (PIN_FLD_DISPUTED). Any dispute amount due to event-level disputes is not touched.

   For debit settlements, the opposite occurs.

2. Calls PCM_OP_BILL_POL_ITEM_VALID_SETTLEMENT to validate the changes.

   You can customize the validation criteria for PCM_OP_BILL_POL_ITEM_VALID_SETTLEMENT.

   For information on customizing the opcode, see "Customizing Item-Level Settlements".

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

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

   - Whether the field contains a POID, indicating that PCM_OP_AR_ITEM_SETTLEMENT is being called by PCM_OP_AR_BILL_SETTLEMENT to complete a bill-level settlement.

     This POID identifies the **/item/settlement** object that PCM_OP_AR_BILL_SETTLEMENT created to store information on the bill-level settlement. This is the **/item/settlement** object that PCM_OP_AR_ITEM_SETTLEMENT associates with each of the settlement events it creates for the bill-level settlement.

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

4. Creates an **/event/billing/settlement/item** object.

   If the item settlement is part of a bill-level settlement, the event object for each settled item is associated with a single, umbrella **/item/settlement** object.

5. Examines the flags in the input flist to determine whether the dispute has any tax implications.

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

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

   Before transferring the funds, PCM_OP_AR_ITEM_SETTLEMENT filters disputed events that were created by event-level disputes and subtracts the amount associated with these events from the PIN_FLD_DISPUTED of the disputed item. What remains is the amount associated solely with the item-level dispute, with no contribution from event-level disputes.

7. If the Due amount after settlement is **0** and there are no other open disputes, PCM_OP_AR_ITEM_SETTLEMENT sets the PIN_ITEM_STATUS field to CLOSED.

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

9. Provides feedback on whether the settlement was successfully created.

   If the settlement creation failed, the opcode indicates the reason for the failure.

BRM creates a G/L ID for item-level settlements. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Item Settlements

Set these fields in the PCM_OP_AR_ITEM_SETTLEMENT input flist when creating item-level settlements:

- PIN_FLD_POID for the item to be settled. The settled amount is applied to the default balance group of the bill unit associated with the item.
- PIN_FLD_AMOUNT for the settlement amount. The value for this field should be negative when crediting and positive when debiting.
- PIN_FLD_CURRENCY for the resource to be settled.

PCM_OP_AR_ITEM_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so forth.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_ITEM_SETTLEMENT.

Flags for Item Settlements

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

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

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

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

Customizing Item-Level Settlements

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

You can customize the validation criteria for PCM_OP_BILL_POL_VALID_SETTLEMENT by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass.

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

Disputing Events

To open a dispute at the event level, use PCM_OP_AR_EVENT_DISPUTE.

This opcode receives a batch of events on the input flist, and returns the disputed **/event/billing/dispute/event** data on the output flist. There can be multiple events in a given batch, and the events to be disputed do not need to belong to the account specified in the PIN_FLD_POID for PCM_OP_AR_EVENT_DISPUTE. The input flist can optionally include a reason code domain and descriptive string that indicates why the dispute is being made.

Caution: PCM_OP_AR_EVENT_DISPUTE does not verify whether there are other disputes against the events it processes. Multiple disputes for the same event can cause conflicts in BRM, so make sure your client application prevents the CSR from logging more than one dispute for an event.

When it receives a batch of events, PCM_OP_AR_EVENT_DISPUTE performs these tasks:

1. It reads the batch, identifying all events to be disputed
2. It examines the flags in the input flist to determine whether the dispute has any tax implications.

   For information on how the opcode performs dispute taxation, see "Including Taxes in the Dispute or Settlement".
3. It creates a single dispute item, **/item/dispute**, for all the disputed events in the batch.
4. For each event to be disputed, it creates a new **/event/billing/dispute/event** object containing the dispute amount.

   The PIN_FLD_ITEM_OBJ field of **/event/billing/dispute/event** points to the dispute item created in step 3.
5. After the **/item/dispute** and **/event/billing/dispute/event** objects are created, it calls PCM_OP_BILL_ITEM_TRANSFER to transfer the dispute amount from the dispute item to the original item that owns the disputed event.

   Note: If the opcode transfer targets a closed item, that item is reopened automatically.
6. It creates a notification event (**/event/billing/dispute/notify**) and passes it to PCM_OP_ACT_USAGE for processing.

   If event notification is enabled in your system, this triggers the reservation of the disputed resources until the dispute is settled (see "Configuring Event Notification for Disputes and Settlements"). For information on reserving resources and customizing resource reservation for disputes, see "Customizing Resource Reservation for Disputes".
7. When it finishes, it provides feedback on whether the dispute was successfully created or, if dispute creation failed, the reason for the failure.

BRM creates a G/L ID for event-level disputes. For information on disputes and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Event Disputes

Set these fields in the input flist when creating event-level disputes:

- PIN_FLD_POID for each of the events to be disputed. These fields form a PIN_FLD_EVENTS array. The disputed amount is applied to the default balance group of the bill unit associated with the **/item** object that the event belongs to.
- PIN_FLD_AMOUNT for the dispute amount. The value supplied for this field should be positive for adjustments that credit the event and negative for adjustments that debit the event.

  You can specify the dispute amount through either PIN_FLD_AMOUNT or PIN_FLD_PERCENT. If you omit both of these fields, BRM disputes the entire amount for each event.
- PIN_FLD_RESOURCE_ID for the disputed resource.

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

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_EVENT_DISPUTE.

Flags for Event Disputes

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

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

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

  Important: Do not set this flag for disputes against unbilled events that use deferred taxation.

- If the PIN_EVENT_ADJUST_BATCH is set, the dispute amount will be distributed among the events in the batch on a first-come-first-serve basis until the dispute amount is consumed.
- If the PIN_EVENT_ADJUST_EVENT is set, the dispute amount will be applied individually to each of the events in the batch.

Customizing Resource Reservation for Disputes

Resource reservation for disputes is performed by the PCM_OP_RESERVE_POL_POST_DISPUTE policy opcode, which reserves resources equivalent to the dispute amount for as long as the dispute is active. PCM_OP_RESERVE_POL_POST_DISPUTE enables you to perform custom processing of the reservation (**/reservation** object) that it creates for the dispute.

When PCM_OP_RESERVE_POL_POST_DISPUTE receives a dispute notification event (**/event/billing/dispute/notify**) from PCM_OP_ACT_USAGE, it reads the dispute item POID from the input flist and obtains all associated dispute events. From the balance impact arrays for each dispute event, it reads:

- PIN_FLD_ACCOUNT_OBJ: to obtain the account or accounts affected by the dispute
- PIN_FLD_AMOUNT: to obtain the balance impact amounts
- PIN_FLD_RESOURCE_ID: to identify the resource type (currency or non-currency)

PCM_OP_RESERVE_POL_POST_DISPUTE then creates a **/reservation** object for each distinct entry in the balance impact arrays and stores the dispute item in the PIN_FLD_SESSION_OBJ field of the **/reservation** object. The output flist provides a pass/fail status to PCM_OP_AR_EVENT_DISPUTE and returns the POID for each **/reservation** object it creates.

You can customize PCM_OP_RESERVE_POL_POST_DISPUTE to notify third parties of the dispute. For example, you can customize PCM_OP_RESERVE_POL_POST_DISPUTE to notify a collection agency of any disputes so that they can postpone collection efforts for the disputed item until settlement. Or, if you have an automatic customer confirmation service in place, you can customize PCM_OP_RESERVE_POL_POST_DISPUTE to notify that service, which can, in turn, inform the customer that the dispute has been created.

You can also customize PCM_OP_RESERVE_POL_POST_DISPUTE by determining whether PCM_OP_ACT_USAGE will call it at all. In most cases, this opcode should be part of the dispute process. However, if you determine that you do not need resource reservation even to ensure accurate credit treatment for prepaid customers, you can omit it.

To configure BRM to run PCM_OP_RESERVE_POL_POST_DISPUTE, you must enable event notification. See "Configuring Event Notification for Disputes and Settlements".

Settling Disputed Events

To settle a dispute at the event level, use PCM_OP_AR_EVENT_SETTLEMENT. This opcode debits or credits the resources of an event to settle a dispute against that event.

Note: You cannot settle an event more than once. If PCM_OP_AR_EVENT_SETTLEMENT settles an event that has more than one open dispute, it settles all the disputes for that event.

PCM_OP_AR_EVENT_SETTLEMENT receives dispute item POID in the input flists and retrieves the associated dispute events. It returns the settled **/event/billing/settlement/event** data on the output flist. The input flist can optionally include a reason code domain and descriptive string that indicates why the settlement is being made.

When it receives a batch of events, PCM_OP_AR_EVENT_SETTLEMENT performs these tasks:

1. It verifies that, for backdated settlements, the settlement date falls after the dispute date.
2. It examines the flags in the input flist to determine whether the settlement has any tax implications.

   For information on how the opcode performs dispute taxation, see "Including Taxes in the Dispute or Settlement".
3. It searches for all dispute events whose PIN_FLD_ITEM_OBJ field points to the dispute item and creates a single settlement item, **/item/settlement**, for all these events.
4. For each dispute event to be settled, it creates a new **/event/billing/settlement/event** object containing the denied amount.

   The PIN_FLD_SESSION_OBJ field of /**event/billing/settlement/event** points to the corresponding dispute event.
5. After the **/item/settlement** and **/event/billing/settlement/event** objects are created, it calls PCM_OP_BILL_ITEM_TRANSFER to transfer the denied amount from the settlement item to the original item against which the settlement was filed and recalculate the Due of the original item accordingly.

   Note: If PCM_OP_AR_EVENT_SETTLEMENT transfer targets a closed item, that item will be reopened automatically.

   To prevent misallocation of resources during settlement, PCM_OP_AR_EVENT_SETTLEMENT calculates the combined dispute amount for the all of the dispute events with respect to the original disputed item. This amount is associated solely with the event-level dispute being settled. It has no contribution from any item-level disputes or any event-level disputes other than the one being settled. It is this amount that PCM_OP_BILL_ITEM_TRANSFER works with when transferring the item.
6. It creates a notification event (**/event/billing/settlement/notify**) and passes to PCM_OP_ACT_USAGE for processing.

   If event notification is enabled in your system, this triggers the release of the dispute **/reservation** object, indicating that the event has been settled (see "Configuring Event Notification for Disputes and Settlements"). For information on reserving resources and customizing resource reservation for disputes, see "Customizing Resource Reservation for Settlements".
7. When it finishes, PCM_OP_AR_EVENT_SETTLEMENT provides feedback on whether the settlement was successfully created or, if settlement creation failed, the reason for the failure.

BRM creates a G/L ID for event-level settlements. For information on settlements and G/L IDs, see "Assigning G/L IDs for a Dispute or Settlement".

Fields You Should Include in the Input Flist for Event Settlements

Set these fields in the PCM_OP_AR_EVENT_SETTLEMENT input flist when creating event-level settlements:

- PIN_FLD_ITEM_OBJ for the dispute item.
- PIN_FLD_AMOUNT for the settlement amount. The value supplied for this field should be positive for settlements that credit the event and negative for settlements that debit the event.

  You can specify the settlement amount through the PIN_FLD_AMOUNT field or omit the field altogether. If you omit this field, BRM settles the entire amount for each event.
- PIN_FLD_RESOURCE_ID for the resource to be settled.

PCM_OP_AR_EVENT_SETTLEMENT accepts a variety of other fields used to define tax treatment, settlement time, reason code, and so forth.

- For information on tax treatment, see "Including Taxes in the Dispute or Settlement".
- For information on reason codes, see "Including Reason Codes in the Dispute or Settlement".
- For information on any remaining fields, see the input flist specification for PCM_OP_AR_EVENT_SETTLEMENT.

Flags for Event Settlements

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

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

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

  Important: Do not set this flag for settlements against unbilled events that use deferred taxation.

- If the PIN_EVENT_ADJUST_BATCH is set, the settlement amount will be distributed among the events in the batch on a first-come-first-serve basis until the settlement amount is consumed.

- If the PIN_EVENT_ADJUST_EVENT is set, the settlement amount will be applied individually to each of the events in the batch.

Customizing Resource Reservation for Settlements

Resource reservation processing for settlements is performed by the PCM_OP_RESERVE_POL_POST_SETTLEMENT policy opcode, which releases resources reserved when an event-level dispute is created. This opcode enables you to perform custom processing of the reservation (**/reservation** object) that it releases during the settlement.

When it receives a settlement notification event (**/event/billing/settlement/notify**) from PCM_OP_ACT_USAGE, PCM_OP_RESERVE_POL_POST_SETTLEMENT reads the dispute item POID from the input flist and searches for all reservation objects that have the dispute item in the PIN_FLD_SESSION_OBJ field. For each of these **/reservation** objects, PCM_OP_RESERVE_POL_POST_SETTLEMENT updates the PIN_FLD_RESERVATION_STATUS field to indicate that the reservation is released and to create a timestamp for the reservation release. The PCM_OP_RESERVE_POL_POST_SETTLEMENT output flist provides a pass/fail status to PCM_OP_AR_EVENT_SETTLEMENT and can optionally generate a descriptive message concerning its success or failure.

You can customize PCM_OP_RESERVE_POL_POST_SETTLEMENT to notify third parties of the settlement; for example a collection agency or an automatic confirmation service.

You can also customize PCM_OP_RESERVE_POL_POST_SETTLEMENT by determining whether PCM_OP_ACT_USAGE will call it at all. If you included dispute reservation as part of the dispute process, PCM_OP_RESERVE_POL_POST_SETTLEMENT should also participate in the settlement process. If you do not, **/reservation** objects created by dispute reservation will not be released on settlement.

To configure BRM to run PCM_OP_RESERVE_POL_POST_SETTLEMENT, you must enable event notification. See "Configuring Event Notification for Disputes and Settlements".

Including Taxes in the Dispute or Settlement

You specify an optional tax treatment for disputes and settlements through flags in the input flist. For information, see:

- Flags for Bill Disputes
- Flags for Bill Settlements
- Flags for Item Disputes
- Flags for Item Settlements
- Flags for Event Disputes
- Flags for Event Settlements

Tax Processing for Disputes

The dispute opcodes calculate taxes when the dispute is created. Before calculating taxes, they validate that the item due is not zero. If it is zero, the dispute is not allowed.

If the input flist flags the dispute or settlement as taxable, the dispute opcode either includes or excludes the tax:

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

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

Tax Processing for Settlements

The settlement opcodes calculates taxes when the settlement is created. Before calculating taxes, they validate that the settlement amount is less than or equal to the disputed amount. If the settlement amount is greater, the settlement is not allowed.

If the original bill item uses taxation, the tax is considered tax inclusive or tax exclusive, depending on the amount of the dispute:

- If the disputed amount is partially granted, the settlement amount is considered to be tax-exclusive. The amount due on the bill item is increased by the net denied amount and the amount due on the cycle tax item is increased by the denied tax amount.
- If the disputed amount is completely granted (there is no denied amount), the settlement amount is considered to be tax-inclusive. The amount due on the bill item does change for taxes.

> **Note:** If the disputed amount is completely denied, the amount due on the bill item is increased by the net amount and the amount due on the tax item is increased by the tax amount.

If the input flist flags the settlement as taxable, the settlement opcode determines whether the tax method is tax now or tax deferred.

- If the tax method is tax now, the settlement opcodes use information from the event associated with the original bill item to calculate taxes and apply the amount immediately with the settlement amount.
- If the tax method is tax deferred, taxes are contained in the tax item associated with the dispute. In such cases, the opcode does the following to handle the taxes:
  - Calls PCM_OP_BILL_TAX_EVENT to calculate the net and tax amount from the total settlement amount. If a denied amount exists, it then calculates the net and tax amount from the total denied amount.
  - Creates the **/event/billing/settlement/tax_event** event to contain the denied tax amount.

- Transfers the denied net amount to the original item and the denied tax amount to the tax item.

Including Reason Codes in the Dispute or Settlement

Reason codes explain why a dispute is being opened or settled. In typical implementations, the BRM client application provides a selectable set of reasons appropriate to the event type. You choose a reason that matches the situation or, in some cases, compose a new reason. The client application populates the input flist for the opcode with this information, and BRM records the reason in the dispute or settlement event object (for example, **/event/billing/dispute/item** or **/event/billing/settlement/event**). The flist fields that provide reason code information are PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID. These fields are optional.

For general information about reason codes, see "String Manipulation Functions" in *BRM Developer's Reference*. For information about setting up reason codes for adjustments, see "Working with Reason Codes for Adjustment, Disputes, and Settlements".

Assigning G/L IDs for a Dispute or Settlement

The G/L ID for an dispute or settlement event is assigned like other pre-rated events in the BRM system. The PCM_OP_ACT_POL_SPEC_GLID policy opcode assigns the G/L ID for the adjustment. For information on preconfiguring the adjustment event to modify the **/config/map_glid** configuration object, see "Assigning G/L IDs to Nonrated Events" in *BRM Collecting General Ledger Data*.

7 Configuring Write-Offs and Write-Off Reversals

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) write-offs and the write-off reversal feature. It also tells you how to configure BRM for write-offs and write-off reversals and how to enable automatic write-off reversals for receipt of payment.

For more information on payments and accounts receivable adjustments, see the following topics:

- About Payments
- About Accounts Receivable

For information on writing off accounts, bills, or bill items, see the Customer Center Help.

For information on reversing payments, see the Payment Tool Help.

For information on customizing write-off reversal rules, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

About Write-Offs

A write-off removes an accounts receivable (A/R) amount that your company considers unrecoverable because the customer will never pay. CSRs can write off an A/R account with or without its nonpaying child subordinates, a bill, or a bill item that meets certain conditions.

For write-offs, you *always* write off the entire account, bill, or bill item.

Important:

- Before you write off an account or bill unit, ensure that all items have been billed and that no pending or unbilled items for the account or bill unit remain.
- To ensure that you can reverse a write-off on an account, the account status must be **inactive** before you write off the account.

About Write-Off Reversals

Write-off reversals enable you to re-allocate payments to accounts and bill units that were written off due to unpaid balances. Unpaid bills and item charges are generally written off as unrecoverable debt, but this feature enables the amount written off to be recovered and allocated to open bills and items of an account.

You reverse the write-off on an account or bill unit by calling the write-off reversal opcodes.

For more information, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

About Automatic Write-Off Reversals during Payment Collection

You can configure BRM to automatically reverse a write-off on an account or bill unit when a payment is received for a written-off account or bill unit. BRM verifies that the write-off flag and write-off item are set in the account's **/profile/writeoff** object. If so, the following occurs:

1. The write-off flag in the account's **/profile/writeoff** object is set to **2**, which is the reversed state.
2. All written-off bills and bill items are opened so the funds can be allocated.
3. The **/event/billing/writeoff/billinfo** balance amount is transferred to the original bill or bill item and then set to **0**.
4. The paid bills and items are closed.
5. The write-off flag in the account's **/profile/writeoff** object is reset to **1**, which is the written-off state.

Important: To ensure that you can reverse a write-off on an account, the account status must be **inactive** before you write off the account.

For information about customizing write-off reversal rules, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

About Reversing a Payment Applied to a Written-Off Account or Bill Unit

When a payment that was applied to a written-off account or bill unit is reversed, the previously paid balance is returned to the account balance and then written off again. This enables the original written-off amount to be reestablished as unrecoverable debt.

Table 7–1 shows an example where multiple payments are received for an account with a $100 write-off amount.

Payment 1 is $40 (underpayment), so the $100 write-off amount is reversed and $60 is written off.

Payment 2 is $100 (overpayment); therefore, the $60 write-off amount is reversed, and $40 excess (credit) amount is left unallocated in the account.

Table 7–1 Reversing Payment Applied to a Written-Off Account

Transaction	Due Amount	Unrecovered Debt	Recovered Debt	Bank Deposit
Account balance	100	NA	NA	NA
Initial write-off	-100	100	NA	NA
Write-off reversal	100	NA	-100	NA
Payment	-40	NA	NA	40
Re–write-off	-60	NA	60	NA
Write-off reversal	60	NA	-60	NA

Table 7–1 (Cont.) Reversing Payment Applied to a Written-Off Account

Transaction	Due Amount	Unrecovered Debt	Recovered Debt	Bank Deposit
Payment Reversal	40	NA	NA	-40
Re–write-off	-100	NA	100	NA
Net Result	**0**	**100**	**0**	**0**

Important: If a subsequent payment is received during this operation, or if an A/R action is performed, the account may not be restored to its previous state. This occurs if any open unallocated items are in the account at the time of the reversal.

If Payment 1 is reversed, the write-off is *not* reversed because there is no longer a written-off amount on the account: instead, a credit in the account balance remains. To avoid this scenario, customize the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to handle the allocation and write-off. Or allocate the credit balance to any open bills or bill items, and then manually write off the account.

For information about customizing write-off reversal rules, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

About Overpayment and Underpayment Allocation

After a payment is posted to a written-off account or bill unit, the handling of the bill or bill items is determined by the amount of the payment: exact payment, overpayment, or underpayment. Exact payments are allocated to the appropriate bill and bill items, which are closed after being paid, and the account's write-off flag is *not* reset to write-off.

Overpayments and underpayments are allocated according to the rules defined in your business policies:

- In general, if the payment amount is *more than* the amount written off, the paid bills and bill items are closed and the remaining amount is left unallocated in the account. The account or bill unit is left open and the account's write-off flag is *not* reset to write-off.
- If the payment amount is *less than* the amount written off, the payment is fully allocated, and the open bills and bill items are closed again. The remaining amount is written off and the account's write-off flag is reset to write-off.
- If an underpayment fails and you reverse it, both the amount that was written off (due to the underpayment) and the amount that was removed by the payment are returned to the account or bill unit. This restores the account balance to its original state so the entire amount can be written off again. This corresponds to the initial write-off amount for the account or bill unit.

  For example, if your company writes off a $50 unpaid balance on an account or bill unit and 6 months later receives a payment of $45 for that account or bill unit, the $50 write-off is returned to the account or bill unit so that the payment can be applied toward the balance. After the payment amount is subtracted, the remaining $5 balance is written off again.

  Later, if the $45 payment fails, the following steps occur when you reverse the payment:

1. The $5 write-off amount is returned to the account balance.
2. The $45 balance that was removed by the initial payment is returned to the account balance.
3. The $50 account balance is written off again.

- If any open unallocated items are in the account at the time of the reversal, the re–write-off on the account does not occur. You must first allocate and close the open items, or customize the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to allocate and close the open items, before performing the reversal.

  For information on customizing write-off reversal rules, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

Overview of the Write-Off Reversal Process

The following steps occur when a payment is made for a written-off amount. It also describes the steps that occur if the original payment must be reversed and the paid amount must be written off again:

1. An account or bill unit contains an unpaid balance that was written off as unrecoverable, and a payment is received for the written-off amount.
2. If the write-off reversal feature is enabled, the write-off on the account or bill unit is reversed.
3. The written-off balance is transferred back to the account or bill unit.
4. The payment is applied to the account or bill unit and all paid-off bill items are closed.
5. Any remaining balance on the account or bill unit (for an underpayment) is written off again.
6. If the payment fails, a reversal of the payment is submitted.
7. BRM verifies that the original payment was applied to the written-off account or bill unit.
8. The second write-off on the account or bill unit is reversed, if one exists (when the write-off reversal payment was an underpayment).
9. Any bill items closed by the original payment are reopened and the amount is transferred back to the account or bill unit.
10. The original payment is reversed.
11. After the payment reversal, if the account or bill unit has a balance due, the amount is written off again to restore the account balance to **0**, and the account or bill unit status is reset to write-off.

Configuring BRM for Write-Off Reversals

Configuring the write-off reversal functionality involves performing the following tasks:

- Defining Reason Codes for Write-Off Reversals
- Mapping G/L IDs to Write-Off Reversal Events
- Enabling Automatic Write-Off Reversals during Payment Collection
- Specifying the Level of Write-Offs for Reversals

Defining Reason Codes for Write-Off Reversals

You define additional reason codes in the **reasons**.*locale* file for the write-off reversal event and the re–write-off of an account. You use the Credit Reasons domain (version 8) for write-off reversal reasons, and you should map the reason code to the **/event/billing/writeoff** G/L ID, as shown in the following sample:

```
DOMAIN = "Reason Codes-Credit Reasons" ;
STR
     ID = 4 ;
     VERSION = 8 ;
     STRING = "Write-off for Auto-writeoff reversal feature" ;
     EVENT-GLID
       "/event/billing/writeoff"     110;
     EVENT-GLID-END
END
```

Note: If you add your own reason codes to the **reasons.***locale* file, you should use IDs above 100,000.

To define reason codes for write-off reversals, you edit the **reasons.en_US** sample file in the *BRM_Home***/sys/msgs/reasoncodes** directory. You then use the **load_localized_strings** utility (see "load_localized_strings" in *BRM Developer's Guide*) to load the contents of the file into the **/strings** objects.

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

```
load_localized_strings reasons.locale
```

Note: If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.

Caution: The **load_localized_strings** utility overwrites the existing **/config/map_glid** object. If you are updating this object, you cannot load new G/L ID maps and reason code scopes only. You must load complete sets of data each time you run the **load_localized_strings** utility. This is also true when using the **/strings** object, but only if you specify the **-f** parameter. Otherwise, the **load_localized_strings** utility appends the new data to the object.

For information on loading the **reasons.***locale* file, see "Loading Localized or Customized Strings" in *BRM Developer's Guide*. For information on creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

Mapping G/L IDs to Write-Off Reversal Events

You can track funds you recover from write-off reversals in your general ledger (G/L) system. In addition, you can track the following events that occur if the payment fails and must be reversed:

- If the payment was an underpayment, the write-off reversal event that occurred for the second write-off, which was required to write off the unpaid balance on the account
- The reversal of the original payment that failed
- The re–write-off of the debt on the account

Depending on your G/L configuration, you can create single or separate G/L entries to track the reversed written-off amount and associated tax amount in respective G/L accounts.

You map custom G/L IDs for write-off reversal events by editing and loading the **reasons.***locale* file. This file contains the reason code definition that is assigned to the G/L ID of the **/event/billing/writeoff** event.

To load a customized **reasons.***locale* file into the BRM database, use the **load_ localized_strings** utility. This utility loads the event-to-G/L ID mapping into a **/config/map_glid** object in BRM. The G/L ID -to-event mapping is defined in the **pin_ glid** file.

You map custom G/L IDs when you define your reason codes for the write-off reversal event. See "Defining Reason Codes for Write-Off Reversals".

For more information about general ledger data, see "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data*.

Enabling Automatic Write-Off Reversals during Payment Collection

You can configure BRM to reverse write-offs on accounts and bill units automatically when payments are received for written-off amounts. This enables the payment to be posted to the accounts and bill units and the written-off amount to be recovered immediately during payment processing.

You enable this feature by modifying a field in the **ar** instance of the **/config/business_ params** object created during BRM installation. As part of payment allocation and billing, BRM reads this object to determine whether write-off reversals can occur automatically when payments are received.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To enable automatic write-off reversals during payment collection:

1. Run the following command, which creates an editable XML file for the **BusParamsAR** parameter class:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

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

2. Search the XML file for the following line:

   ```
   <AutoWriteOffReversal>disabled</AutoWriteOffReversal>
   ```

3. Change **disabled** to **enabled**.

Caution: BRM uses the XML in this file to overwrite the existing **/config/business_params** object for the **ar** instance. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

---

4. Run the following command, which loads the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_AR.xml
   ```

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

5. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using testnap" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

6. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

For information on how to customize the write-off level to search for and to be available for reversal during payment processing, see "Writing Off Debts and Reversing Write-Offs with Your Custom Application".

Specifying the Level of Write-Offs for Reversals

By default, BRM is configured to perform write-off reversals at the account level when payments are received.

You can configure BRM to perform write-off reversals at the bill-unit level when payments are received for a written-off bill unit. When you configure BRM to perform write-off reversals at the bill-unit level, BRM reverses write-offs at both the bill-unit level and the account level.

You enable this feature by modifying a field in the **ar** instance of the **/config/business_params** object created during BRM installation. As part of payment allocation and billing, BRM reads this object to determine whether write-off reversals can occur on an account or bill unit when payments are received.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For more information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To specify the level of write-offs for reversals:

1. Go to the *BRM_Home*/**sys/data/config** directory.

2. Run the following command, which creates an editable XML file from the **ar** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

This command creates the XML file named **bus_params_AR.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.

3. Open the **bus_params_AR.xml.out** file.
4. Search for the following line:

```
<WriteOffLevel>account</WriteOffLevel>
```

5. To specify the level of write-offs for reversals to occur at the bill-unit level, change **account** to **billinfo**.
6. Save the file as **bus_params_AR.xml**.
7. Go to the *BRM_Home*/**sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the **/config/business_params** object:

```
pin_bus_params PathToWorkingDirectory/bus_params_AR.xml
```

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

Caution: BRM uses the XML in this file to overwrite the existing **ar** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM A/R configuration.

Note: To run this command from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using the **testnap** utility. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.
10. Stop and restart the CM. For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Writing Off Debts and Reversing Write-Offs with Your Custom Application

For background information, see "About Writing Off Bad Debt".

Write-offs remove A/R amounts that your company determines will never be paid. Write-off reversals allocate payments to accounts, bills, bill units, and items that have been previously written off. Write-offs are a standard feature of your BRM system. The ability to reverse a write-off is an optional feature that you must enable and configure. For information on enabling and configuring write-off reversals, see "Enabling Automatic Write-Off Reversals during Payment Collection" and "Configuring

Write-Offs and Write-Off Reversals".

About Initiating Write-Offs

Use the following opcodes to perform write-offs:

- PCM_OP_AR_ACCOUNT_WRITEOFF. Writes off all A/R bill units for an account and each bill unit's subordinate nonpaying children. To have your client application initiate an A/R account write-off, it should call this opcode and the input flist should specify the account POID.
- PCM_OP_AR_BILLINFO_WRITEOFF. Writes off an A/R bill unit for an account. To have your client application perform an A/R bill unit write-off, it should call this opcode with the bill unit specified in the input flist.
- PCM_OP_AR_BILL_WRITEOFF. Writes off a bill. To have your client application perform a bill write-off, it should call this opcode with the bill specified in the input flist.
- PCM_OP_AR_ITEM_WRITEOFF. Writes off a bill item. To have your client application perform an item write-off, it should call this opcode with one or more items specified in the input flist.

Important:

- Before you write off an account or bill unit, ensure that all items have been billed and that there are no pending items for the account or bill unit.
- To ensure that you can reverse a write-off on an account, the account status must be inactive before you write off the account.

How BRM Performs Write-Offs

Though you can initiate a write-off using opcodes for the account, bill unit, bill, or item level, most of any write-off is actually performed through the PCM_OP_AR_ITEM_WRITEOFF opcode. This opcode is either:

- Called by the PCM_OP_AR_BILLINFO_WRITEOFF and PCM_OP_AR_BILL_WRITEOFF opcodes when you perform an account-level, bill-unit level, or bill-level write-off

  Note: PCM_OP_AR_ACCOUNT_WRITEOFF, PCM_OP_AR_BILLINFO_WRITEOFF, and PCM_OP_AR_BILL_WRITEOFF perform several initial steps before calling PCM_OP_AR_ITEM_WRITEOFF. For more information, see "About Account Write-Offs", "About Bill Unit Write-Offs", and "About Bill Write-Offs".

- Called independently when you perform an item-level write-off

  Note: PCM_OP_AR_ITEM_WRITEOFF performs write-offs on pending items with nonzero balances only.

PCM_OP_AR_ITEM_WRITEOFF does the following:

1. Prepares the charges for the **/item** object (for example, **/item/misc**).

2. Calls the PCM_OP_BILL_POL_ITEM_VALID_WRITEOFF policy opcode to validate the changes.

   For information on how to customize write-off validation, see "Customizing Write-Off Validation".

3. Creates the new write-off item (**/item/writeoff**).

   For an account or bill write-off, the item total is the sum of all item charges in the account or bill.

4. Creates the **/event/billing/writeoff/item** object.

   The balance impact of the write-off event is the sum of all amounts due for all open items in the bill unit.

   - If an account is written off, an **/event/billing/writeoff/account** object is created for the account and an **/event/billing/writeoff/billinfo** object is created for each bill unit.
   - If a bill is written off, an **/event/billing/writeoff/bill** object is created.

5. If you have specified to store the tax amount of the write-off in a separate event, PCM_OP_AR_ITEM_WRITEOFF creates one of the following event objects:

   - **/event/billing/writeoff/tax_billinfo**
   - **/event/billing/writeoff/tax_bill**
   - **/event/billing/writeoff/tax_item**

   For information on tax treatments for write-offs, see "About Taxes for Write-Offs".

6. Calls the PCM_OP_BILL_TRANSFER opcode to transfer funds to the A/R bill.

   Multiple transfer events (**/event/billing/item/transfer**) are created to move the credit from the write-off item to all the open items for the bill unit.

7. If the item has no due or disputed amount, PCM_OP_AR_ITEM_WRITEOFF closes the item.

8. Writes the modified **/item** object.

When this process is complete, all the open items, including the write-off item, are closed. The G/L ID for the account-level write-off is assigned by calling the PCM_OP_ACT_POL_SPEC_GLID policy opcode, which assigns the G/L ID for the write-off event (**/event/billing/item/writeoff**).

If write-off reversal is enabled, it stores the write-off item in the write-off profile object of the account.

Important:

- Before you write off an account or bill unit, ensure that all items have been billed and that no pending or unbilled items for the account or bill unit remain.
- To ensure that you can reverse a write-off on an account, the account status must be inactive before you write off the account.

About Account Write-Offs

To write off an account, the PCM_OP_AR_ACCOUNT_WRITEOFF opcode writes off *all* the bill units associated with the account.

This opcode performs the following error checks before performing the item-level write-offs for the account:

- Ensures the due amount is not zero. Bill units with zero due are fully paid and do not need to be written off.
- Ensures that there are no partially allocated items (for example, a refund must be allocated before the write-off is performed).

PCM_OP_AR_ACCOUNT_WRITEOFF then calls PCM_OP_AR_BILLINFO_WRITEOFF to write off all the open items for each bill unit of the account.

The **/event/billing/writeoff/account** event is created to record the account write-off. The **/event/billing/writeoff/billinfo** event is created to record each bill unit write-off, and **/event/billing/writeoff/tax_billinfo** is created for each bill if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as PIN_BILL_WRITEOFF_TAX in the input flist.

Your custom application should call PCM_OP_AR_ACCOUNT_WRITEOFF, not PCM_OP_AR_BILLINFO_WRITEOFF, to write off *all* bill units in an account.

About Bill Unit Write-Offs

To write off a bill unit, PCM_OP_AR_BILLINFO_WRITEOFF performs the following error checks before performing the item-level write-offs for the bill unit:

- Ensures the due amount is not zero. Bill units with zero due are fully paid and do not need to be written off.
- Ensures that there are no partially allocated items (for example, a refund must be allocated before the write-off is performed).

PCM_OP_AR_BILLINFO_WRITEOFF then calls PCM_OP_AR_ITEM_WRITEOFF to write off each of the bill items in the bill unit.

When a bill unit is written off, a write-off item (**/item/writeoff**) is created. The total for this item is the sum of the balance due from all the open items for the A/R bill units and its nonpaying child subordinate bill units. This is the balance impact of the write-off. The write-off decreases the account balances and the transfer event moves the credit from the write-off item to the items being written off.

The **/event/billing/writeoff/billinfo** event is created to record a bill unit write-off, and the **/event/billing/writeoff/tax_billinfo** event is created for the bill unit if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as PIN_BILL_WRITEOFF_TAX in the input flist.

About Bill Write-Offs

To write off a bill, PCM_OP_AR_BILL_WRITEOFF performs the following error checks before performing the item-level write-offs for the bill:

- Ensures the due amount is not zero. Bills with a zero due are fully paid and do not need to be written off.

PCM_OP_AR_BILL_WRITEOFF then calls PCM_OP_AR_ITEM_WRITEOFF to write off each of the bill items in the bill.

When a bill is written off, a write-off item (**/item/writeoff**) is created. The write-off decreases the account balances associated with the specified bill, and the transfer event moves the credit from the write-off item to the item being written off.

The **/event/billing/writeoff/bill** event is created to record a bill write-off, and **/event/billing/writeoff/tax_bill** is created for the bill if a separate event for tax is required. A separate tax event can be generated by passing the PIN_FLD_FLAG as 1 in the input flist.

Flags You Should Use for Write-Offs

Set flags in the PIN_FLD_FLAGS field for the input flist when creating write-offs. You can include these flags in any of the write-off opcodes.

If the PIN_BILL_WRITEOFF_TAX flag is set, BRM creates a separate write-off event for taxes.

About Taxes for Write-Offs

To write off tax, BRM selects all the open items. For all open items, PCM_OP_AR_ITEM_WRITEOFF selects all events that point to these items, except for the tax events. After the events are selected, it calculates the sum of the total initial net and tax amount and groups the events based on tax rate. The write-off process starts with the lowest tax rate first and continues with the next lowest rate until it reaches the total write-off amount. The total write-off amount (VAT included) is the total due amount of the bill.

The following types of events are created for a write-off with tax:

- A write-off event holding the total net amount of the write-off with a G/L ID for the net amount
- A tax write-off event holding the VAT amount of the write-off with a G/L ID for tax amount
- Transfer events to close all the remaining open items of the bill

You can use a CM **pin.conf** entry to select the distribution algorithm for taxes.

Customizing Write-Off Validation

You use the PCM_OP_BILL_POL_VALID_WRITEOFF policy opcode to customize how the item is validated for a write-off. By default:

- The item must be open.
- The write-off amount must be less than or equal to the amount due.

The PCM_OP_BILL_POL_VALID_WRITEOFF policy opcode is called by PCM_OP_AR_ITEM_WRITEOFF. It checks the validity of field values before processing. Field values either pass or fail. If one field fails, the entire operation fails.

You can customize the validation criteria for the PCM_OP_BILL_POL_VALID_WRITEOFF policy opcode by modifying the policy source file. Changing a result from PIN_BOOLEAN_FALSE to PIN_BOOLEAN_TRUE enables the specified field value to pass. Changing a result from PIN_BOOLEAN_TRUE to PIN_BOOLEAN_FALSE causes the specified field value to fail.

About Initiating Write-Off Reversals

Write-off reversals are initiated when PCM_OP_PYMT_COLLECT encounters a payment for an account or bill unit that has been written off. PCM_OP_PYMT_COLLECT runs automatically as part of the payment collection process or can be called manually. PCM_OP_PYMT_COLLECT calls the write-off reversal opcode to perform the write-off reversal.

Note: Automatic write-off reversals are an optional feature of BRM and must be enabled. For information on enabling automatic write-off reversals, see "Enabling Automatic Write-Off Reversals during Payment Collection".

The following opcodes perform the actual reversal:

- PCM_OP_AR_REVERSE_WRITEOFF: This opcode reverses the write-off and opens the original bill items so that the payment can be allocated.

  Note: Write-off reversals should only be performed automatically; you should not call PCM_OP_AR_REVERSE_WRITEOFF directly.

- PCM_OP_BILL_POL_REVERSE_PAYMENT: This policy opcode writes off an account that was previously written off, but which then received a payment that reversed the original write-off. For example, if a payment that reversed the write-off of an account is determined to be fraudulent, the account will again be written off.

How BRM Reverses Write-Offs

Write-off reversals are performed by PCM_OP_AR_REVERSE_WRITEOFF. This opcode is called by PCM_OP_PYMT_COLLECT.

Important: If you want a separate event to record the tax amount, set the PIN_BILL_WRITEOFF_TAX flag in the input flist.

Note: Write-off reversals should only be performed automatically; you should not call PCM_OP_AR_REVERSE_WRITEOFF directly.

PCM_OP_AR_REVERSE_WRITEOFF performs the following tasks:

1. Validates that the item in the input flist is a written-off item.

   If there are no written-off items in the input flist, PCM_OP_AR_REVERSE_WRITEOFF calls the PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode to retrieve the items that must be reversed.

2. Opens the originally written-off items so that funds can be allocated.
3. Creates a write-off reversal item (**/item/writeoff_reversal**).
4. Creates a write-off reversal event (**/event/billing/writeoff_reversal**) that points to the write-off reversal item.
5. Updates the account's balance.
6. (Account-level write-off reversal only) Sets the value of the **/profile/writeoff** object's PIN_FLD_ITEM_OBJ field to NULL.

   PCM_OP_AR_REVERSE_WRITEOFF also sets the PIN_FLD_WRITEOFF_INFO flag in this object to the reversed state (**2**). If the payment is not equal to or greater than the sum of the written-off items, this flag is reset to the write-off state (**1**).

How BRM Reverses a Write-Off Reversal

When BRM receives a payment for a written-off account or bill unit, it reverses the account or bill unit write-off and then allocates the payment. If that payment must later be reversed (for example, if it fails financially), BRM must then write off the account or bill unit a second time.

This activity is performed by the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode. This policy opcode is called by PCM_OP_BILL_REVERSE_PAYMENT, and the input flist for the policy opcode includes any flags passed by the standard opcode.

Important: If any open unallocated items are present in the account or bill unit at the time of the reversal, the second write-off on the account does not occur. You can either allocate and close the open items before performing the reversal or customize this policy opcode to perform the task.

The PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode performs the following operations if the PIN_FLD_STATUS value on the input flist is PIN_PYMT_WRITEOFF_SUCCESS and if the transaction ID of the payment to be reversed is valid and applied to a written-off amount.

1. Write off the account or bill unit again (if the original payment was an underpayment).
2. Assign a reason code for the second write-off.
3. Set the value of the PIN_FLD_FLAGS field to PIN_BILL_WRITEOFF_TAX to generate a separate tax event (if necessary), and assign a reason code for the tax event.

The PIN_FLD_INHERITED_INFO substruct in the output flist passes additional information to the calling opcode. If necessary, PCM_OP_BILL_REVERSE_PAYMENT returns any information in this substruct to the calling opcode.

Customizing Write-Off Reversals

You can customize two aspects of write-off reversals:

- Use the PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode to customize the rules that govern how BRM performs write-off reversals.

  See "Customizing the Rules for Performing Write-Off Reversals".

- Use the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to customize the way that BRM processes payments that reverse account write-offs but that are, in turn, reversed. Payments of this sort result in a second write-off of the account.

  See "Customizing Reversal of Payments Allocated to Written-Off Accounts".

Customizing the Rules for Performing Write-Off Reversals

You can customize the PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode to implement additional rules for performing write-off reversals and allocating funds to written-off accounts and bill units. For example, you can perform the write-off reversal only for accounts that were written off during a specified time period or for a minimum amount.

The PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode is called by PCM_OP_AR_REVERSE_WRITEOFF during the write-off reversal process. This policy opcode supplies a list of write-off items that require reversal if that list is not provided by any other means. The input flist for the policy opcode includes any flags passed by the standard opcode. The PCM_OP_AR_POL_REVERSE_WRITEOFF policy opcode retrieves the write-off reversal items from the **/profile/writeoff** object.

Important: If any open unallocated items are present in the account or bill unit at the time of the reversal, the second write-off on the account does not occur. You can either allocate and close the open items before performing the reversal, or customize this policy opcode to perform the task.

To enable automatic write-off reversal, see "Enabling Automatic Write-Off Reversals during Payment Collection".

Customizing Reversal of Payments Allocated to Written-Off Accounts

You can implement special processing for reversing payments that were applied to written-off accounts. To implement this processing, you customize the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to perform the following functions:

- Assign custom reason codes.
- Generate a tax event based on custom business policies.
- *Not* reverse write-offs when a payment is applied.
- Allocate any open items in the account before performing the reversal, before the reversal can occur. This is required if multiple payments are received, and a subsequent payment results in an overpayment.

  For example, if a $100 write-off amount is present for an account and a $40 payment is received, followed by a $90 payment, a $30 credit is left unallocated on the account.

  Because write-offs are valid only for debit items, if you reverse the $40 payment, BRM creates a $40 *debit* open item and a $30 *credit* open item, and the result would be a $10 unallocated balance. This is not possible; therefore, the unallocated credit of $30 must be allocated before the re–write-off can be performed.

  To enable automatic allocation, customize the PCM_OP_BILL_POL_REVERSE_PAYMENT policy opcode to call PCM_OP_BILL_ITEM_TRANSFER, and allocate the open credit amount to the available debit items before performing the re–write-off.

8

Retrieving A/R Information for Your Custom Application

This chapter describes how to retrieve accounts receivable (A/R) data by using Oracle Communications Billing and Revenue Management (BRM) opcodes.

About Retrieving A/R Information

You can retrieve various kinds of balance information, such as the following:

- Current balances for items and bills
- A list of open and pending bill items
- A list of bills
- A list of adjusted items

To retrieve balance information, use the following opcodes:

- **PCM_OP_AR_GET_ACCT* opcodes**: These opcodes retrieve the current balance for open and pending bill items associated with all the **/billinfo** objects in the account or for a specified **/billinfo** object. In the latter case, you specify the **/billinfo** object in the input flist.
- **PCM_OP_AR_GET* opcodes**: These opcodes retrieve the current balance for open and pending bill items associated with the specified **/billinfo** object. Specify the **/billinfo** object in the input flist.

Note: To retrieve information for all of an account's **/billinfo** objects using PCM_OP_AR_GET*, implement custom code that recursively calls PCM_OP_AR_GET* for each **/billinfo** object. This technique produces results similar to using PCM_OP_AR_GET_ACCT*, but it involves the overhead of creating a customization.

For information on viewing A/R information in Customer Center, see the Customer Center Help.

Finding a Bill

To find a bill, use the PCM_OP_BILL_FIND opcode. You can use this opcode to search for all **/bill** objects instead of using the PCM_OP_SEARCH opcode and the PCM_OP_STEP_SEARCH opcode.

Note: PCM_OP_BILL_FIND does not perform authentication.

- If you specify an array of bill numbers, PCM_OP_BILL_FIND returns the POID of the **/bill** object.
- If you specify the results array, the fields in the results array are returned.
- If you do not specify the results array, PCM_OP_BILL_FIND returns the entire contents of the storable object.

Finding a Bill Unit

To find the bill units (**/billinfo** objects) for an account, use the PCM_OP_BAL_GET_ACCT_BILLINFO opcode. This opcode 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 PCM_OP_BAL_GET_ACCT_BILLINFO to get contact and billing information for an account. The input flist contains the account POID. The output flist contains:

- Contact information from the **/account** object, including the account's contact name, address, and phone number.
- The POIDs, names, and payment methods of all the **/billinfo** objects associated with that account. If more than one **/billinfo** object is returned, the default **/billinfo** array contains a PIN_FLD_FLAGS entry of **1**. The PIN_FLD_FLAGS entry for each of the other **/billinfo** arrays is **0**.

For information about bill units, see "About Bill Units".

Finding a Balance Group and Its Balances

To retrieve a **/balance_group** POID and, optionally, the balances it contains, use the PCM_OP_BAL_GET_BALANCES opcode. This opcode returns the **/balance_group** and **/billinfo** POIDs for an account or service. Using the PIN_FLD_BALANCES array, you can also direct this opcode to return any or all of the balances contained in the **/balance_group** object.

The opcode finds the **/balance_group** POID by using the event end time and the **/service** object's PIN_FLD_TRANSFER_LIST array. For more information, see "Transferring Services between Balance Groups by Using Custom Client Applications".

You can specify the validity period for which to retrieve sub-balances for the given balance group by setting the PIN_FLD_FLAGS field to one of the following values:

- PIN_BAL_GET_BARE_RESULTS (0x1): This flag returns all sub-balances for the current cycle that are currently valid.
- PIN_BAL_GET_ALL_BARE_RESULTS (0x4): This flag returns all sub-balances for the current cycle, including those that are expired.
- PIN_BAL_GET_BASED_ON_PERIOD (0x10): This flag returns all sub-balances whose validity period falls within or overlaps a given period, specified by the PIN_FLD_START_T and PIN_FLD_END_T fields.

  For example, if the period is June 1 through June 30, the opcode retrieves all sub-balance that were valid for any time between June 1 and June 30, whether their validity period begins before June 1 or ends after June 30.

- PIN_BAL_GET_WITHIN_PERIOD (0x16): This value returns only the sub-balances whose start and end times fall entirely within the given time period specified by the PIN_FLD_START_T and PIN_FLD_END_T fields.

  For example, if the period is June 1 through June 30, the opcode retrieves only sub-balances that start on or after June 1 and end on or before June 30.

If no balance is available for the specified period, PCM_OP_BAL_GET_BALANCES returns 0.

To return the balances that were granted by a specific product or discount, specify the POID of the granting product or discount in the PIN_FLD_GRANTOR_OBJ field.

When this opcode is called by the Activity Facilities Module (FM) opcodes or the Resource Reservation Framework (RRF), it searches the **/reservation_list** objects in In-Memory Database (IMDB) Cache and the **/balance_group** object in the database to get the balance. It performs the following steps to get the balances:

1. Reads the PIN_FLD_RESERVED_AMOUNT field in the **/balance_group** object for the specified account.
2. Reads the PIN_FLD_AMOUNT field from the PIN_FLD_BALANCES array of the **/reservation_list** object for the balance group.
3. Adds the amounts from the **/balance_group** object PIN_FLD_RESERVED_AMOUNT field and the **/reservation_list** object PIN_FLD_BALANCES array and returns that amount.

This opcode returns balances that are configured to start when first impacted if their validity periods have not yet been set. When a balance's validity period has not been set, the PIN_FLD_VALID_FROM_DETAILS and PIN_FLD_VALID_TO_DETAILS fields are returned for that balance. These fields store three values in separate bits: the mode of the validity start and end times (such as first usage or relative), the relative offset unit, and the number of offset units in the relative period.

If best pricing is configured, this opcode makes backup copies and resets the balance cache at the beginning of each calc-only rerating operation during the best pricing analysis phase.

Important:

- A read/write transaction must be open before calling PCM_OP_BAL_GET_BALANCES. Otherwise, the object is locked, and the read operation returns an error.
- You can call this opcode within a read-only operation, but the PIN_FLD_READ_BALGRP_MODE field in the input flist must be set to PIN_BAL_READ_BALGRP_CALC_ONLY.

Finding a Balance Group and Service for Bill Units

To retrieve a balance group and service for all the bill units in an account or for a single bill unit, use the PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC opcode.

To retrieve the balance group and service for a single bill unit, use the PCM_OP_BAL_GET_BAL_GRP_AND_SVC opcode.

These opcodes return the **/balance_group** and **/service** object POIDs based on the **/account** POID, **/billinfo** POID, and event end time (PIN_FLD_END_T) passed in the input flist. The balance groups for a specified bill unit do not have to be associated with a service yet to be returned.

If the CALC_ONLY flag is set, PCM_OP_BAL_GET_BAL_GRP_AND_SVC reads the balance groups and services from the cache instead of from the database.

The PIN_FLD_FLAGS field in the input flist controls the type of information returned in the output flist. The settings for this flag are:

- **PIN_BAL_GET_SERVICE_DEFAULT (0)**: do not return login name and balance group name (default).
- **PIN_BAL_GET_SERVICE_LOGIN (1)**: Return the login name and balance group name.
- **PIN_BAL_GET_SERVICE_ALIAS_LIST (2)**: Return the alias list for the service.
- **PIN_BAL_GET_DEFAULT_BAL_GRP (3)**: Return the default balance group of the bill unit.
- **PIN_BAL_GET_DEFAULT_GRP_AND_SVC (4)**: Return the default balance group of the bill unit and the default service of the default balance group.
- **PIN_ BAL_GET_ALL_BAL_GRP_AND_SVC (5)**:Return all balance groups, including balance groups not associated with a service, balance groups associated with a service, and the account-level balance group.

Note: You can also pass in flags to get the balance group name and service login aliases.

Balance groups not yet associated with a service are returned when the PIN_FLD_FLAGS value is not set or is set to **0**, **1**, or **2**. If the balance group returned is the account-level balance group, the service object field in the output flist is NULL. If the balance group returned is *not* the account-level balance group, there is no service object field in the output flist. This enables balance groups not associated with a service to be displayed in Customer Center.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC over PCM_OP_BAL_GET_BAL_GRP_AND_SVC because it gives you the additional flexibility of retrieving balance groups and services for either multiple or single bill units.

For information about balance groups, see "About Balance Groups".

Finding Items

To find items, use the PCM_OP_PYMT_ITEM_SEARCH opcode. This opcode searches for **/item** objects with a variable number of input parameters. It is called by other opcodes for handling adjustments and disputes.

Functionality:

- This search takes a variable number of arguments to restrict the scope of the search.
- The PIN_FLD_POID field is mandatory and is a dummy POID used for getting the database ID for the search.
- The bill unit (**/billinfo** object) field is optional. This field specifies the bill unit that owns the items. You can use either the **/billinfo** POID or the **/billinfo** ID. If a **/billinfo** is not specified, the **/billinfo** object associated with the account's default balance group is selected.

- You can specify either PIN_FLD_ACCOUNT_OBJ or PIN_FLD_ACCOUNT_NO to narrow the search to a particular account.
- For collective bills, if they are not located in the bill table, this opcode checks the **RejectPaymentForPreviousBill** business parameter.
  - If **RejectPaymentForPreviousBill** is enabled to prevent BRM from accepting the payment, the opcode sends the payment to the suspense payment account.
  - If the **RejectPaymentForPreviousBill** is disabled to prevent BRM from accepting the payments, the opcode will search for the bill in the **history_bills** objects. If the bill is located in the **history_bills** objects, the payment is accepted for the bill with the same POID as the bill being processed. If such a bill is not located in **history_bills** objects, the payment is sent to a suspense payment account.
- If PIN_FLD_STATUS, PIN_FLD_BILL_OBJ, or both are specified, the search matches the items equal to these values.
- PIN_FLD_START_T and PIN_FLD_END_T specify the time range for the search. These arguments narrow the search to items whose PIN_FLD_CREATED_T value falls between the start and end times.
- The PIN_FLD_POID field in the PIN_FLD_ARGS array narrows the search to specific **/item** object types. This is a TYPE_ONLY POID. Different items have different POID object types.

  For example, if you want the search to return only the Cycle Forward and Usage items, create the POID for the PIN_FLD_POID field as shown below:

  ```
  PIN_POID_CREATE(database,
        " '/item/cycle_forward', '/item/usage' " -1, ebufp);
  ```

For information about items, see "How BRM Stores Accounts Receivable Information".

Retrieving a Balance Summary

To retrieve the balance summary from an account, use the PCM_OP_AR_GET_ACCT_BAL_SUMMARY opcode. This opcode retrieves the unapplied, open bill due, pending bill due, and total dispute balances for all the bill units (**/billinfo** objects) in a specified account or for a particular bill unit. For example, Customer Center uses PCM_OP_AR_GET_ACCT_BAL_SUMMARY to retrieve the balance summary of a bill unit or bill units displayed on the **Balance** tab.

You can use the PCM_OP_AR_GET_BAL_SUMMARY opcode to get this information for a single bill unit.

For both opcodes, you can specify whether BRM returns the balance for:

- The specified bill units (for PCM_OP_AR_GET_BAL_SUMMARY, this is a single bill unit)
- The specified bill units and their nonpaying child bill units (for PCM_OP_AR_GET_BAL_SUMMARY, this is a single bill unit)

For example, in Customer Center, selecting **Include child amounts** for a parent account includes the amounts of its child accounts' nonpaying bill units in the calculation of its balance summary and bills.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_BAL_SUMMARY over PCM_

OP_AR_GET_BAL_SUMMARY because it gives you the additional flexibility of retrieving balance summaries for either multiple or single bill units.

About the Balance Summary Data Retrieved

PCM_OP_AR_GET_ACCT_BAL_SUMMARY retrieves the following data for the specified bill unit:

Note: The data includes the amounts of the bill unit's nonpaying child bill units if you set the appropriate flag.

- Unapplied amount
- Amount due for the open bill
- Amount due for the pending bill
- Total disputed amount
- For PCM_OP_AR_GET_BAL_SUMMARY, the Bill in Progress information when there are pending items and the balance is greater than or equal to 0 because of open disputes against the items.

For more information, see the PCM_OP_AR_GET_ACCT_BAL_SUMMARY and PCM_OP_AR_GET_BAL_SUMMARY output flist specs.

Specifying Search Criteria for Retrieving a Balance Summary

You can use the following input criteria for retrieving the balance:

- The POID of the A/R bill unit object or *paying* **/billinfo**
- The POID of the bill unit for which to retrieve the balance summary
- A flag indicating whether to retrieve the balances for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return bills for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN flag to **0**.
  - To return bills for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN flag to **1**.
- For PCM_OP_AR_GET_BAL_SUMMARY, whether to retrieve Bill in Progress information if there are any pending items but the balance is zero.

  To do so, PCM_OP_AR_GET_BAL_SUMMARY uses the PIN_FLD_ITEM_PENDING_FLAGS flag. If this flag is set to **1** in the output flist, the opcode provides access to Bill in Progress information on the balance summary displayed in Customer Center. If is flag is set to **0**, the opcode does not provide access to Bill in Progress information.

For more information, see the PCM_OP_AR_GET_ACCT_BAL_SUMMARY and PCM_OP_AR_GET_BAL_SUMMARY input flist specs.

Retrieving a List of Bills for a Bill Unit

Use the PCM_OP_AR_GET_ACCT_BILLS opcode to get a list of bills for all the bill units (**/billinfo** objects) in a specified account or for a particular bill unit (see *BRM*

Developer's Reference). For example, Customer Center uses this opcode to list an account's bills in the Bills section of the **Balance** tab.

You can use the PCM_OP_AR_GET_BILLS opcode to get this information for a single bill unit.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_BILLS over PCM_OP_AR_GET_BILLS because it gives you the additional flexibility of retrieving the bill list for either multiple or single bill units.

About the Bill Data Retrieved

PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS retrieve a list of bills for the account's bill units or it retrieves the bill units and their nonpaying child bill units. PCM_OP_AR_GET_BILLS retrieves only the bill unit you specify in the input flist. For each bill, the output includes the following data:

- Current and previous total
- Total for nonpaying child bill units, provided you set the appropriate flag
- Amounts for adjustments, disputes, transfers, write-offs, and so forth

For more information, see the PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS output flist specs.

Specifying the Search Criteria for Retrieving Bills

When you use PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS, you can use the following input criteria for retrieving the list of bills:

- The POID of the account, **/billinfo** object, or **/bill** object

  These opcodes list a specific bill if the **/bill** object POID is passed in the input flist. If you specify an account POID for PCM_OP_AR_GET_ACCT_BILLS, the opcode retrieves the bills for all the bill units in the account.

- The POID of the A/R bill unit

  Note:

  - For PCM_OP_AR_GET_ACCT_BILLS, this field is mandatory if the input flist specifies the **/billinfo** or **/bill** POID instead of the account POID.
  - For PCM_OP_AR_GET_BILLS, this field is mandatory unless you include the POID of the A/R account.

- The POID of the A/R account. This returns the default bill unit for the A/R account

  Note: This field is not available for PCM_OP_AR_GET_ACCT_BILLS.

- A flag indicating whether to retrieve the balances for only the given bill unit or for the given bill unit and its nonpaying child bill units:

  - To return balances for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return balances for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.

  For example, in Customer Center, selecting **Include child amounts** for a parent account sets this flag, and the data returned by the opcode includes the amounts of the child accounts' nonpaying bill units in the calculation of its balance summary and bills.

- The status of the bills to retrieve. If the PIN_FLD_STATUS flag is set to:
  - **2**: The opcode retrieves open bills.
  - **4**: The opcode retrieves closed bills.
  - **6**: The opcode retrieves all bills.
- The number of bills to retrieve. If you use PCM_OP_AR_GET_ACCT_BILLS to retrieve bills from all the bill units in an account, this is the number of bills the opcode retrieves from each bill unit. It is *not* the total number of bills to retrieve from the account unless the account has only one bill unit.
- The start and end times for the bills to retrieve.
- The sort order, ascending or descending. If you use PCM_OP_AR_GET_ACCT_BILLS to retrieve bills from all the bill units in an account, this is the sort order the opcode applies to the bills within each bill unit, *not* across all the bill units.

For more information, see the PCM_OP_AR_GET_ACCT_BILLS and PCM_OP_AR_GET_BILLS input flist specs.

Retrieving A/R Items that Apply to a Bill Unit

Use the PCM_OP_AR_GET_ACCT_ACTION_ITEMS opcode (see *BRM Developer's Reference*) to get a list of A/R items for all the bill units (**/billinfo** objects) in a specified account or for a particular bill unit. For example, Customer Center uses this opcode to list items that relate to adjustments, payments, reversals, disputes, settlements, refunds, and write-offs for an account or a given bill unit of an account.

The A/R item types returned can be:

- **/item/adjustment**
- **/item/dispute**
- **/item/payment**
- **/item/refund**
- **/item/writeoff**

Items associated with the A/R items, such as, **/item/payment/reversal** and **/item/settlement**, are returned in the PIN_FLD_RELATED_ACTION_ITEM_OBJ field.

You can use the PCM_OP_AR_GET_ACTION_ITEMS opcode to get this information for a single bill unit.

If you are developing the interface between BRM and a third-party client application, you should typically favor PCM_OP_AR_GET_ACCT_ACTION_ITEMS over PCM_OP_AR_GET_ACTION_ITEMS because it gives you the additional flexibility of retrieving the A/R items for either multiple or single bill units.

About the Item Data Retrieved

PCM_OP_AR_GET_ACCT_ACTION_ITEMS and PCM_OP_AR_GET_ACTION_ITEMS return the following data:

Note: PCM_OP_AR_GET_ACCT_ACTION_ITEMS returns an array. If the PIN_FLD_POID field is the account POID in the input flist, each element in the array is one of the listed fields for a **/billinfo** object in the account. Otherwise, the returned fields are for the requested single bill info.

- The **/billinfo** and A/R bill unit object (PIN_FLD_BILLINFO_OBJ and PIN_FLD_AR_BILLINFO_OBJ) that the item belongs to.
- The account's primary contact information (PIN_FLD_NAMEINFO array), such as company, first name, last name, middle name, and salutation.
- The number of A/R items that were found (PIN_FLD_THRESHOLD) *only* if this count exceeds the threshold specified in the input. If the specified threshold is **not** exceeded, all the A/R items found are returned.
- Either the A/R items for *only* the given bill unit or the A/R items for the given bill unit *and* its nonpaying child bill units, depending on the flag setting in the input flist
- For payments and reversals (that apply):
  - The POID of the reversal item (PIN_FLD_RELATED_ACTION_ITEM_OBJ)
  - The revision time when the payment was reversed; that is, when the reversal was posted (PIN_FLD_POSTED_T)
- For disputes and settlements (that apply):
  - The POID of the settlement item (PIN_FLD_RELATED_ACTION_ITEM_OBJ)
  - The revision time when the dispute was settled; that is, when the settlement was posted (PIN_FLD_POSTED_T)
  - The bill item to which the dispute or settlement is made (PIN_FLD_RELATED_BILL_ITEM_OBJ)
- For PCM_OP_AR_GET_ACTION_ITEMS only:
  - For event-level adjustments, disputes, and settlements, this opcode returns a PIN_FLD_AGGREGATE_AMOUNTS array for each event that makes up the adjustment, dispute, or settlement item. This array contains the aggregated amounts for each of the resources associated with the event.
  - For adjustments, disputes, and settlements, this opcode indicates whether the adjustment, dispute, or settlement was created at the event level or the item level. PCM_OP_AR_GET_ACTION_ITEMS also indicates whether the adjustment, dispute, or settlement affects single or multiple resources.

For possible return values of each A/R action retrieved, refer to the fields contained in the PIN_FLDS_RESULTS array in the output flist specification.

Specifying Search Criteria for Retrieving Items

PCM_OP_AR_GET_ACCT_ACTION_ITEMS and PCM_OP_AR_GET_ACTION_ITEMS take as input:

- The POID of the A/R bill unit (PIN_FLD_AR_BILLINFO_OBJ).
- The POID of the bill unit related to the A/R items.

  When retrieving A/R items for nonpaying child bill units, the **/billinfo** object is that of the paying bill unit. Otherwise, it is the **/billinfo** object of the specified bill.
- The POID of the bill unit for which to retrieve A/R items (PIN_FLD_AR_BILLINFO_OBJ). This field is required only when retrieving A/R items for a specific, nonpaying child bill unit.
- A flag indicating whether to retrieve the A/R items for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return A/R items for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return A/R items for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- The maximum number, or threshold, of A/R items to retrieve.

  Note:

  - The threshold is set in the **Threshold** tab of Customer Center Preferences as the default number of A/R items displayed for an account in the A/R Actions section of the **Bill Details** panel.
  - The threshold field is valid only for an A/R bill unit (that is, the paying bill unit of a parent account).
  - If you use PCM_OP_AR_GET_ACCT_ACTION_ITEMS to retrieve A/R items from all the bill units in an account, this is the maximum number of A/R items the opcode retrieves from each bill unit. It is not the total number of A/R items retrieved from the account unless the account has only one bill unit.

- The action-item type to retrieve (for example, **/item/adjustment**) specified as a comma-delimited string

  Note: When retrieving only payments, both payments and reversals are specified. When retrieving only disputes, both disputes and settlements are specified.

- The time range for which to retrieve A/R items. The start date is inclusive and the end date is exclusive.
- The amount range. The from and to amounts are inclusive.
- The amount indicator for whether to retrieve credits or debits for the given range
- The item status. You can query based on a status of open, closed, or reversed (valid for payments only).

  Note: There is no *pending* status for A/R items.

Retrieving a List of Bill Items for a Bill Unit

To retrieve a list of bill items for a bill unit (**/billinfo** object), use the PCM_OP_AR_GET_BILL_ITEMS opcode.

Customer Center, for example, uses this opcode to list the usage items, cycle items, and custom items for a given bill in the Item Charges section of the **Bill Details** panel.

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

Depending on the value of the following Connection Manager (CM) **pin.conf** entry, the charge and discount for the items is populated in the output flist of the PCM_OP_AR_GET_BILL_ITEMS opcode. The value for this entry can be either **0** or **1**. The default is **0**.

```
- fm_ar skip_displaying_charge_and_discount_for_item 1
```

If the value specified is **1**, the charge and discount for the items is not populated in the output flist of PCM_OP_AR_GET_BILL_ITEMS.

Note: This opcode is used by Customer Center to display the item details in the **Balance** tab. Setting this flag to **1** skips the query to get the charge and discount from the events belonging to the item, thereby improving performance.

About the Bill Item Data Retrieved

PCM_OP_AR_GET_BILL_ITEMS returns:

- The bill items for only the given bill unit, if PIN_FLD_INCLUDE_CHILDREN is set to **0**
- The bill items for the given bill unit and its nonpaying child bill units, if PIN_FLD_INCLUDE_CHILDREN is set to **1**
- The bill items for the nonpaying child bill units, if PIN_FLD_INCLUDE_CHILDREN is set to **2**
- The number of bill items that were found (PIN_FLD_THRESHOLD) *only* if this count exceeds the threshold specified in the input. If the specified threshold is *not* exceeded, all the bill items found are returned.

For each bill item retrieved, the fields in the PIN_FLDS_RESULTS array are returned. For the fields contained in this array, see the PCM_OP_AR_GET_BILL_ITEMS output flist specification.

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. See "Enabling and Disabling General Ledger Collection in BRM" in *BRM Collecting General Ledger Data*.

Specifying the Number of Items to Retrieve in a Search

PCM_OP_AR_GET_BILL_ITEMS uses step-search. The step size (the number of items it tries to retrieve simultaneously) is determined by the following CM **pin.conf** entry:

```
- fm_bill item_search_batch 10000
```

Specifying Search Criteria for Bill Items

PCM_OP_AR_GET_BILL_ITEMS takes as input:

- The POID of the A/R bill unit or *paying* **/billinfo** object (PIN_FLD_AR_BILLINFO_OBJ)
- The POID of the bill object related to the bill items

  When PCM_OP_AR_GET_BILL_ITEMS retrieves bill items for nonpaying child bill units, the bill object is that of the paying **/billinfo** object. Otherwise, it is the bill object of the specified bill unit.
- The POID of the **/billinfo** object (PIN_FLD_BILLINFO_OBJ) for which to retrieve bill items. This field is required only when retrieving bill items for a specific, nonpaying child bill unit.
- A flag indicating whether to retrieve the bill items for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return bill items for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return bill items for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
  - To return bill items for the nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **2**.
- The service device ID, or phone number, or login name to retrieve the SERVICE_OBJ for the services with matching ALIAS or LOGIN_NAME. SERVICE_OBJ is used to retrieve the associated bill items.
- The maximum number, or threshold, of bill items to retrieve.

  This is set in the **Threshold** tab of Customer Center Preferences as the default number of bill items displayed for an account in the Item Charges section of the **Bill Details** panel.
- The time range in which to retrieve bill items. The start date is inclusive and the end date is exclusive.
- The amount range. The from and to amounts are inclusive.
- The amount indicator for whether to retrieve credits or debits for the given range
- The item status. You can query based on a status of open, closed, or pending.

Retrieving Details about a Specific A/R Item or Bill Item

You can retrieve details for an A/R item or bill item in a bill unit using either of two opcodes:

- **PCM_OP_AR_GET_ITEMS**: Use this opcode to retrieve the details of an A/R item or bill item for a bill unit. For adjustments, disputes, and settlements, this opcode also retrieves the aggregated amount of each resource for the events associated with the item (**/item/dispute**, **/item/adjustment**, and so forth). PCM_OP_AR_GET_ITEMS provides details for both currency and non-currency resources.

  Customer Center uses PCM_OP_AR_GET_ITEMS to display currency and non-currency details for A/R actions associated with a specific bill item (adjustments, disputes, settlements, payments, refunds, write-offs, and so forth). It also uses PCM_OP_AR_GET_ITEMS to provide details for an A/R action item even if only the non-currency resources were impacted by the action.

- **PCM_OP_AR_GET_ITEM_DETAIL**: Use this opcode to retrieve the details of an A/R item or bill item for a bill unit. This opcode does not return as much information as PCM_OP_AR_GET_ITEMS. For example, it does *not* return the aggregated resources for dispute events. More importantly, it does *not* return details on non-currency resources for adjustments, disputes, or settlements unless the item has both a currency *and* non-currency component.

You should base your choice of opcode on how much information you want to retrieve for the A/R items and bill items.

About the Item Data Retrieved

PCM_OP_AR_GET_ITEMS calls PCM_OP_AR_GET_ITEM_DETAIL to get the details of the A/R item or bill item. When returning details for adjustments, disputes, or settlements, it collects details for:

- Item-level adjustments, disputes, and settlements (**/event/billing/dispute/item**, **/event/billing/adjustment/item**, and so forth)
- Event-level adjustments, disputes, and settlements (**/event/billing/dispute/event**, **/event/billing/adjustment/event**, and so forth)

Depending on the contents of the input flist, PCM_OP_AR_GET_ITEMS takes one of the following actions:

- If the input flist contains the POID of a transfer event, PCM_OP_AR_GET_ITEMS calls PCM_OP_AR_GET_ITEM_DETAIL to get the details of the transfer and performs no further action. For information on what this opcode returns for transfer events, see "About the Item Detail Data Retrieved".
- If the input flist contains the POID of a bill item or an A/R action such as an adjustment, dispute, or settlement, PCM_OP_AR_GET_ITEMS calls PCM_OP_AR_GET_ITEM_DETAIL to get the details of the A/R action or bill item. It also calls the PCM_OP_AR_RESOURCE_AGGREGATION opcode to calculate and return the aggregated amount of each resource for any event-level adjustments, disputes, and settlements.

  For A/R actions, it determines whether the action was initiated at the item level or event level. If the action was initiated at the event level, it retrieves both currency and non-currency resources for the event.

  For information on PCM_OP_AR_RESOURCE_AGGREGATION, see "Retrieving Details on Available Resources".

PCM_OP_AR_GET_ITEMS returns an array of details for the A/R action or bill item. The **transfers_into** array provides the details of all items that have a currency or non-currency impact for the A/R item or bill item specified in the input flist. These details include the aggregated resources for each event.

For adjustments, disputes, and settlements, the details for each A/R action include a flag in the transfer array that indicates whether the action was initiated at the event level or item level:

- PIN_FLD_ADJUSTMENT_TYPE: Set to **0** for item-level adjustments or **1** for event-level adjustments.
- PIN_FLD_DISPUTE_TYPE: Set to **0** for item-level disputes or **1** for event-level disputes.

- PIN_FLD_SETTLEMENT_TYPE: Set to **0** for item-level settlements or **1** for event-level settlements.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for any adjustments, disputes, or settlements. If this flag is set to **0**, the action affected one resource only. If the flag is set to **1**, the action affected several different resources; for example, multiple currencies or both a currency and a non-currency resource such as free minutes.

For the fields in the transfer array, see the output flist specification.

About the Item Detail Data Retrieved

PCM_OP_AR_GET_ITEM_DETAIL retrieves details of activities that had a currency impact for the specified A/R item or bill item. If the input flist contains the POID for the transfer event, that POID is used to retrieve the item-level details from the transfer event binary large object (BLOB) file. PCM_OP_AR_GET_ITEM_DETAIL returns an array of the bill items the transfer event affected and the input POID.

- The **transfers_into** array returns the details of all the items that received some currency amount from the item specified in the input flist. For example, when a customer service representative (CSR) opens a dispute for a subscription fee, money is transferred into the dispute item from the subscription bill item (the cycle forward item). When the cycle forward item is passed in the input flist, the details of all the disputes that received money from this bill item are returned in the **transfers_into** array.
- The **transfers_out** array returns the details of all those items that transferred some currency amount to the item specified in the input flist. For example, when a customer pays for service usage, money is transferred from the payment item to the bill item (usage item). When the usage item is passed in the input flist, the details of all the payments that transferred money to this bill item are returned in the **transfers_out** array.

For adjustments, disputes, and settlements, the details for the A/R action include a flag in the transfer array that indicates whether the action was initiated at the event level or item level:

- PIN_FLD_ADJUSTMENT_TYPE: Set to **0** for item-level adjustments or **1** for event-level adjustments.
- PIN_FLD_DISPUTE_TYPE: Set to **0** for item-level disputes or **1** for event-level disputes.
- PIN_FLD_SETTLEMENT_TYPE: Set to **0** for item-level settlements or **1** for event-level settlements.

PCM_OP_AR_GET_ITEM_DETAIL determines how to set this flag by checking to see whether the event is stored in an **/event/billing/***AR_action_type***/item** object or an **/event/billing/***action_type***/event** object. The opcode returns the following information:

- **Item-level adjustments, disputes and settlements**: Returns a transfer array that includes details of the adjustment, dispute, or settlement item in the PIN_FLD_ EVENT_OBJ field. In this case, the only event PCM_OP_AR_GET_ITEM_DETAIL looks at is the single event associated with the **/item** object. For example, it looks at the single **/event/billing/dispute/item** event associated with the **/item/dispute** object.
- **Event-level adjustments, disputes and settlements**: Returns a transfer array that includes a PIN_FLD_EVENTS array with the details of all the events that make up

the adjustment, dispute, or settlement. In this case, PCM_OP_AR_GET_ITEM_DETAIL looks at all the events identified in the **/item** object.

The details for each A/R action also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for any adjustments, disputes, or settlements. If this flag is set to **0**, the action affected one resource only. If the flag is set to **1**, the action affected several different resources; for example, multiple currencies or both a currency and a non-currency resource such as free minutes.

In addition to retrieving the other details of the adjustment, dispute, or settlement, PCM_OP_AR_GET_ITEM_DETAIL determines whether there is a tax reversal already calculated for the event. If so, it adds this information to the PIN_FLD_TAX field in the output flist.

Note: Tax reversals are calculated at adjustment, dispute, or settlement event creation time only if you have set up tax now as the tax method for adjustments, disputes, and settlements in the CM **pin.conf** file. For more information, see "Configuring the Default Tax Method for Account-Level Adjustments".

The details for each A/R action also include a PIN_FLD_RESOURCE_IMPACTED flag to indicate whether the action affects a single resource (**0**) or multiple resources (**1**).

For the fields in the transfer array, see the output flist specification.

Specifying Search Criteria for Retrieving Items

PCM_OP_AR_GET_ITEMS and PCM_OP_AR_GET_ITEM_DETAIL take as input:

- The POID of the item object or the POID of the transfer event. If the POID is for a transfer event, all other values are ignored.
- The POID of the disputed item (PIN_FLD_RELATED_BILL_ITEM_OBJ) if the bill item is disputed. This field is mandatory if the item object is a dispute, but you can also include this field for other types of item objects.
- The POID of the settlement item or the reversal item (PIN_FLD_RELATED_ACTION_ITEM_OBJ) if the item is for a dispute or payment *and* a dispute settlement or payment reversal exists.

Retrieving Dispute Details for a Bill Unit

You can retrieve dispute details for a bill unit (**/billinfo** object) using either of two opcodes:

- To return a complete set of dispute details, use the PCM_OP_AR_GET_DISPUTE_DETAILS opcode. This opcode retrieves the details of all event-level and item-level disputes for a bill unit and the aggregated amount of each resource for the dispute events associated with a dispute item (**/item/dispute** object). Customer Center uses PCM_OP_AR_GET_DISPUTE_DETAILS to get the list of disputes and available resources for a bill unit.

  See "Retrieving a Full Set of Dispute Data".
- To return a limited set of dispute details, use the PCM_OP_AR_GET_DISPUTES opcode. This opcode retrieves details of event-level and item-level disputes for a bill unit. This opcode does not return as much information as PCM_OP_AR_GET_DISPUTE_DETAILS. For example, it does *not* return the aggregated resources for dispute events.

See "Returning a Limited Set of Dispute Data".

You should base your choice of opcode on how much information you want to retrieve for the disputes.

Retrieving a Full Set of Dispute Data

PCM_OP_AR_GET_DISPUTE_DETAILS calls PCM_OP_AR_GET_DISPUTES to get the details of each dispute. It collects details for both item-level disputes (**/event/billing/dispute/item**) and event-level disputes (**/event/billing/dispute/event**).

When it has the details for each dispute, PCM_OP_AR_GET_DISPUTE_DETAILS creates an input flist that contains the A/R event POID of each dispute and passes it to the PCM_OP_AR_RESOURCE_AGGREGATION opcode. PCM_OP_AR_RESOURCE_AGGREGATION calculates and returns the aggregated amount of each resource for each dispute event. Resources can be currency or non-currency. For information on PCM_OP_AR_RESOURCE_AGGREGATION, see "Retrieving Details on Available Resources".

Data Retrieved by PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES returns an array of disputes. This array includes transfer arrays identifying the **/item/dispute** object for each event-level and item-level dispute. In addition, the arrays for the event-level disputes include an aggregated resource subarray.

Each dispute includes a PIN_FLD_DISPUTE_TYPE flag in the transfer array to indicate whether it was filed as an event-level dispute or an item-level dispute:

- If this flag is set to **0**, the dispute is an item-level dispute.
- If this flag is set to **1**, the dispute is an event-level dispute.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for the disputes. If this flag is set to **0**, the action affected one resource only. If the flag is set to **1**, the action affected several different resources; for example, multiple currencies or both a currency and a non-currency resource such as free minutes.

For information on the fields in the dispute array, see the PCM_OP_AR_GET_DISPUTES output flist specification.

Specifying Search Criteria for PCM_OP_AR_GET_DISPUTE_DETAILS

PCM_OP_AR_GET_DISPUTES takes as input:

- The POID of the bill unit that has a dispute. If the account object is specified, the default account-level bill unit is selected.
- The POID of the A/R bill unit or paying **/billinfo** object (PIN_FLD_AR_BILLINFO_OBJ)
- A flag indicating whether to retrieve the dispute details for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return dispute details for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return dispute details for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- A status flag indicating whether to retrieve open disputes, closed (settled) disputes, or both. If the PIN_FLD_STATUS flag is set to:

- **2**: The opcode retrieves open disputes.
- **4**: The opcode retrieves settled disputes.
- **6**: The opcode retrieves all disputes, open or settled.

Returning a Limited Set of Dispute Data

PCM_OP_AR_GET_DISPUTES calls the PCM_OP_AR_GET_ACTION_ITEMS opcode to get a list of disputes for the bill unit and passes each disputed item to the PCM_OP_AR_GET_ITEM_DETAIL opcode, which gathers the details on the disputed items. PCM_OP_AR_GET_DISPUTES then creates an output flist that provides the information transferred by PCM_OP_AR_GET_ITEM_DETAIL, including flags to indicate whether the dispute is an event-level dispute or an item-level dispute.

For open disputes, PCM_OP_AR_GET_DISPUTES retrieves unsettled disputes; for resolved disputes, it retrieves settled disputes. This opcode is called directly by Customer Center or by PCM_OP_AR_GET_DISPUTE_DETAILS.

Data Retrieved by PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES returns an array of disputes. Each dispute includes a PIN_FLD_DISPUTE_TYPE flag in the transfer array to indicate whether it was filed as an event-level dispute or an item-level dispute:

- If this flag is set to **0**, the dispute is an item-level dispute.
- If this flag is set to **1**, the dispute is an event-level dispute.

The details also include a PIN_FLD_RESOURCE_IMPACTED flag in the transfer array for the disputes. If this flag is set to **0**, the action affected one resource only. If the flag is set to **1**, the action affected several different resources; for example, multiple currencies or both a currency and a non-currency resource such as free minutes.

For the fields in this array for each element representing a dispute, see the output flist specification of this opcode.

Specifying Search Criteria for PCM_OP_AR_GET_DISPUTES

PCM_OP_AR_GET_DISPUTES takes as input:

- The POID of the bill unit that has a dispute. If the account object is specified, the default account-level bill unit is selected.
- The POID of the A/R bill unit or *paying* **/billinfo** object (PIN_FLD_AR_BILLINFO_OBJ)
- A flag indicating whether to retrieve the dispute details for only the given bill unit or for the given bill unit and its nonpaying child bill units:
  - To return dispute details for the specified bill unit, set the PIN_FLD_INCLUDE_CHILDREN field to **0**.
  - To return dispute details for the specified bill unit and nonpaying child bill units, set the PIN_FLD_INCLUDE_CHILDREN field to **1**.
- A status flag indicating whether to retrieve open disputes, closed (settled) disputes, or both. If the PIN_FLD_STATUS flag is set to:
  - **2**: The opcode retrieves open disputes.
  - **4**: The opcode retrieves settled disputes.
  - **6**: The opcode retrieves all disputes, open or settled.

Retrieving Details on Available Resources

Use the PCM_OP_AR_RESOURCE_AGGREGATION opcode to retrieve the available resources for the events in a bill item. This opcode calculates the aggregated amount of each resource for an event. If there is an adjustment, dispute, or settlement associated with the event, PCM_OP_AR_RESOURCE_AGGREGATION also calculates the aggregated dispute, adjustment, or settlement amount for each resource. The resource types can include currency resources, non-currency resources, or both. The output flists contains the available amounts for opening new adjustments, disputes, or settlements for each resource in the output flist.

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. This opcode is also called by the PCM_OP_AR_GET_DISPUTE_DETAILS opcode and the PCM_OP_AR_GET_ITEMS opcode.

About the Resource Data Retrieved

PCM_OP_AR_RESOURCE_AGGREGATION returns an array that lists the following information for each event:

Note: If PCM_OP_AR_RESOURCE_AGGREGATION is returning adjustment, dispute, or settlement events, it includes a resource array for all resources in the event, whether or not a particular resource is impacted by the adjustment, dispute, or settlement. For example, given a currency adjustment of an event that has US dollars and free minutes as resources, the opcode returns resource arrays for both resources even though the adjustment did not affect free minutes.

- Resource ID
- Amount of the event
- If the resource was discounted, the discount amount
- If the resource was adjusted, the adjustment amount
- If the resource was disputed, the dispute amount and the amount of the original dispute
- If a settlement occurred, the settlement amount
- The total resource currently available. This is the amount in the PIN_FLD_AMOUNT field minus any discounts, adjustments, disputes (the dispute amount in PIN_FLD_DISPUTED only, *not* the original dispute amount), and settlements.

Each event in the array can have multiple resource arrays, each of which includes the above information. Some of the preceding information is optional. For more information on optional fields, see the output flist specification for this opcode.

Specifying Search Criteria for Retrieving Resource Data

PCM_OP_AR_RESOURCE_AGGREGATION takes as input:

- The POID of the account in which to find the events
- The array of event POIDs for which to calculate resources

Finding Events Associated with an Account

To find all events associated with an account, use the PCM_OP_BILL_POL_EVENT_SEARCH policy opcode.

For filtered searches, this policy opcode supports searching on the following criteria:

- POID of the bill unit (the **/billinfo** object)
- POID of the bill
- POID of the item
- Date ranges
- Amount ranges
- Event type
- Service type
- Resource ID

To specify the number of events to return, use a value for the threshold. If the threshold value is greater than or equal to the number of events, all events are returned. If the threshold value is less than the number of events, the number of events is returned.

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

If the PIN_FLD_FLAGS field in the output flist's event array is set to **1** for a particular event, the event has a non-currency impact, and Customer Center provides access to Usage Details (through Bill Details). The PCM_OP_BILL_POL_EVENT_SEARCH policy opcode also provides feedback on its success or failure through the PIN_FLD_DESCR field of the PIN_FLD_RESULTS array in the output flist.

For a complete list of all fields returned, see the output flist specification for this opcode.

Finding Events Associated with Bill Items

To find events associated with a bill item, use the PCM_OP_BILL_ITEM_EVENT_SEARCH opcode.

This opcode searches the **/event** object for details related to a specific item. This opcode retrieves a list of events for a given item POID and flag.

The PIN_FLD_SEARCH_TYPE field in the PCM_OP_BILL_ITEM_EVENT_SEARCH input flist can be set to one or all of the following fields:

- PIN_ITEM_EVENT_OWNED: Events owned by an item. If PIN_ITEM_EVENT_OWNED is selected, PCM_OP_BILL_ITEM_EVENT_SEARCH returns a list of events whose item POID matches the item POID passed in the input flist.
- PIN_ITEM_EVENT_SPONSORED: Events sponsored by the item. If the PIN_ITEM_EVENT_SPONSORED flag is selected, PCM_OP_BILL_ITEM_EVENT_SEARCH returns a list of events whose item POID in the event subtotal array matches the item POID passed in the input flist.
- PIN_ITEM_EVENT_TRANSFERRED: A/R events that affect the item's total. If the PIN_ITEM_EVENT_TRANSFERRED flag is selected, PCM_OP_BILL_ITEM_

EVENT_SEARCH returns a list of events whose item POID matches POIDs stored in PIN_OBJ_TYPE_EVENT_ITEM_TRANSFER. Lists are returned in separate arrays so that a list can be returned for each selected flag.

In addition to the above criteria, the event search can be narrowed to reflect the type of event and when the event was created.

9 About Transferring Rollover Resources

This chapter describes rollover transfers and explains how to configure Oracle Communications Billing and Revenue Management (BRM) for rollover transfers.

About Allowing Customers to Transfer Rolled-Over Resources

Resources rolled over from one billing cycle to another are usually rolled over within the same account. By using the rollover-transfer feature, you can transfer non-currency rollover resources from one account or service instance to other accounts or service instances. For example, a parent can request to have any unused free minutes roll over to a son's or daughter's account.

BRM can perform rollover transfers in the following ways:

- From one account to other accounts.
- From a service instance in one account to service instances in other accounts.
- From one account to service instances in other accounts.
- From a service instance in one account to other accounts.
- From a service instance in one account to other service instances in the same account.
- Between services, even when the service types differ.

  The sender's service type does not need to match the receiver's service type as long as both services own the same type of resource, such as dollars or minutes.

BRM cannot transfer rolled over resources when:

- A member service shares a balance group with its subscription service.

  See "About Rollover Transfers and Subscription Groups".
- The buckets were created as a result of a previous rollover transfer. Previously rolled over resources cannot be rolled over again.

About the Rollover-Transfer Profile

You specify how to transfer rollovers by using a rollover-transfer profile. The rollover-transfer profile specifies:

- The account or service that owns the resource grant (the sender).
- The list of resources to roll over.
- The accounts or services into which the rollover is transferred (the receiver).

- (Optional) The validity periods in which the rollovers can transfer.

You create, modify, and delete rollover-transfer profiles in Customer Center. See the Customer Center Help.

When you create a rollover-transfer profile, BRM, by default, verifies that the profile does not have overlapping validity periods or receivers for a given service. You can add or modify the verification checks by customizing the PCM_OP_CUST_POL_VALID_PROFILE policy opcode.

You can select any non-currency resource to configure a rollover transfer unless there is a rollover rule defined in a product and the sender has not purchased that product. The sender must own the product in which the rollover rule is defined for the rollover transfer to occur.

How BRM Performs Rollover Transfers

BRM performs rollover transfers by using the PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER opcode for each account or service that has a rollover-transfer profile associated with it.

BRM performs a rollover transfer as follows:

1. The BRM event notification system listens for the following events to occur:
   - **/event/billing/cycle/rollover/monthly**
   - **/event/billing/cycle/rollover_correction**

   See "Configuring Event Notification for Rollover Transfers".
2. When one of the events occurs, the event notification system calls PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER.
3. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER checks the rollover-transfer profile object (**/profile/rollover_transfer**) to make sure it is configured and valid for the resource and the receiver.
4. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER transfers the rollover amount to the receivers' accounts.
5. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER removes the rolled-over amount from the sender's sub-balance and creates new sub-balances for the receivers with the transferred amount.

   The validity of the new sub-balance is based on the accounting cycle of the receiver and is valid for the remainder of the current accounting cycle and the next accounting cycle. The receiver's new sub-balance is not subject to future rollovers and its valid-from date is the end date of the sub-balance that was rolled over.
6. PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER generates the **/event/billing/cycle/rollover_transfer** event, which has the same GLIDs as the balance impacts of the original rollover event.
7. If delayed billing is configured and rolled-over resources were consumed by the sender by delayed events, PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER receives the rollover correction event and adjusts the rollover transfer amount of the receiver accordingly.

Important: You must perform rerating for the receiver if events rated before the rollover correction were consumed from the rollover transfer amount. See "About Rerating the Receiver's Account due to Delayed Billing".

About Rollover Transfers and Subscription Groups

If you offer subscription services, you can transfer rolled-over resources from a subscription service to a member service only if they belong to different balance groups. A member service that shares the same balance group as its subscription service *can receive* a rollover transfer but *cannot send* a rollover transfer.

For more information about subscription services, see "About Subscription Services" in *BRM Managing Customers*.

About Rerating the Receiver's Account due to Delayed Billing

When delayed billing is configured, resources rolled over to the receiver during the initial billing process can be accessed by delayed events of the sender.

After a rollover transfer, if the sender account is rerated and the rollover buckets are adjusted, the receiver's rollover amount is also affected. When a bill is finalized for the sender, BRM generates a rollover correction event to adjust the rollover of the sender if any rolled-over resources were consumed by delayed events. BRM adjusts the rollover transfer amount of the receiver accordingly.

Important: You must manually rerate the receiver's account if events rated before the rollover correction were consumed from the rollover-transfer amount. BRM does not automatically create rerate jobs for the receiver.

Note: The receiver must be rerated if there is rollover correction for the sender, even when the sender is not rerated.

Configuring BRM to Use Rollover Transfers

Before you can use rollover transfers, you must perform the following tasks:

1. Enable rollover transfers in the **/config/business_params** object.

   See "Enabling Rollover Transfers in BRM".

2. Set up event notification for rollover transfers.

   See "Configuring Event Notification for Rollover Transfers".

Enabling Rollover Transfers in BRM

To enable rollover transfers, update the **subscription** parameter instance in the **/config/business_params** object by using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

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

```
pin_bus_params -r  BusParamsSubscription bus_params_subscription.xml
```

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

2. Open the **bus_params_subscription.xml.out** file in a text editor.
3. In the **BusParamsSubscription** section, enable rollover transfers by changing the **RolloverTransfer** tag as follows:

```
<BusParamsSubscription>
...
   <RolloverTransfer>enabled</RolloverTransfer>
</BusParamsSubscription>
```

4. Save the file as **bus_params_subscription.xml**.
5. Go to the *BRM_Home***/sys/data/config** directory and load the change into the **/config/business_params** object by using the following command:

```
pin_bus_params bus_params_subscription.xml
```

> **Note:** To run the utility from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Reading an Object and Fields" in *BRM Developer's Guide*.

   The resulting flist of the **testnap** utility must resemble this example, with the PIN_FLD_PARAM_VALUE field value set to **1**:

```
0 PIN_FLD_POID  POID [0] 0.0.0.1 /config/business_params 9806 0
0 PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR  STR [0] "Business logic parameters for
                         Subscription"
0 PIN_FLD_HOSTNAME  STR [0] "-"
0 PIN_FLD_PARAMS  ARRAY [1] allocated 4, used 4
1 PIN_FLD_DESCR  STR [0] "Parameter to enable or disable Rollover
                          Transfer feature. 1 means enabled."
1 PIN_FLD_PARAM_NAME  STR [0] "rollover_transfer"
1 PIN_FLD_PARAM_TYPE  INT [0] 1
1 PIN_FLD_PARAM_VALUE  STR [0] "1"
```

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

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Configuring Event Notification for Rollover Transfers

To use rollover transfers, you must configure event notification to call the opcode that performs the rollover transfers. BRM uses the **/event/billing/cycle/rollover/monthly** and **/event/billing/cycle/rollover_correction** events to trigger a rollover transfer.

To configure event notification for rollover transfers, do the following:

1. Depending on which BRM features you use, your system may contain one or more configuration files for event notification.

   If your system contains more than one of these files, you must merge their contents into a single file.

   All of the event notification files available in your system are in the *BRM_Home*/**sys/data/config** directory.

2. Add the following entries to the event notification file:

   ```
   # Event notification for rollover transfers
   9069    0    /event/billing/cycle/rollover/monthly
   9069    0    /event/billing/cycle/rollover_correction
   ```

   This configures the **/event/billing/cycle/rollover/monthly** and **/event/billing/cycle/rollover_correction** events to call the PCM_OP_SUBSCRIPTION_TRANSFER_ROLLOVER opcode (opcode ID number 9069) to determine if a rollover transfer can occur.

   See "How BRM Performs Rollover Transfers".

3. (Optional) If necessary, add, modify, or delete entries in your event notification file.

4. (Optional) If necessary, create custom code for event notification to trigger.

5. Load your event notification file into the BRM database's **/config/notify** object by running the **load_pin_notify** utility (see "load_pin_notify" in *BRM Managing Customers*).

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

10 Balance Monitoring

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Balance Monitoring Manager.

Before reading this document, you should be familiar with BRM system architecture, pipeline rating, and event notification. For information, see the following:

- "BRM System Architecture" in *BRM Concepts*
- "About Batch Rating" in *BRM Concepts*
- "Integrating BRM with Enterprise Applications" in *BRM Developer's Guide*

For information about using the balance monitoring interface in Customer Center, see the Customer Center Help.

About Balance Monitoring

You use balance monitoring to enable your customers to monitor the balance of a group of accounts in near real time. Balance monitoring collects the balance impacts for a specified group of accounts and services, and notifies customers when their balance is too high. Customers and CSRs can also access the group's total balance at any time by using a custom client interface.

For example, a family with multiple wireless telephony accounts can create a balance monitor that tracks the entire family's balance and that alerts them when the balance total exceeds $100.

Balance monitoring provides advantages to:

- *Service providers* because it reduces the risk of debt exposure from nonpaying customers. Customers who track their balances are less likely to accrue excessive charges that they cannot pay.
- *Customers* because they can monitor spending habits and are alerted when charges exceed their specified values. They can then reduce usage or inform members of the group that are generating excessive charges.

To create a balance monitor, customers define:

- All account and service balances they want to include in their balance monitor. This group of accounts and services forms a *monitor group*.

  See "About Monitor Groups".
- When to be alerted that the balance total is too high or approaching the maximum.

  See "Alerting Customers When Monitored Balances Cross Limits or Thresholds".

About Monitor Groups

Customers specify which account or service balances they want to track by creating a monitor group. Each monitor group includes the following:

- **The owner**. See "About Monitor Group Owners".
- **All members**. See "About Monitor Group Members".
- **The monitor group type**. See "About Monitor Group Types".

Customers can create monitor groups manually or automatically:

- With manual creation, customers specify each member to include in the monitor group.
- With automated creation, customers only specify an owner and monitor type, and BRM automatically finds and adds members to the monitor group. Customer Center balance monitoring supports automated creation.

See "About Creating and Maintaining Balance Monitors".

About Monitor Group Owners

Monitor group owners have permission to view group balances, add or remove group members, and add or change threshold and credit limit values.

Monitor group owners can be accounts or services and can be anywhere in a hierarchy chain. For example, an owner could be a parent account in an account hierarchy or a paying or nonpaying child account.

About Monitor Group Members

Any of the following can be members of a monitor group:

- **Accounts**. Members of a monitor group can include parent accounts, paying child accounts, and nonpaying child accounts. BRM monitors the balances for all services under a member account. For example, if a member account contains a GSM service and a GPRS service, BRM adds the current balance for both services to the group balance.
- **Services**. Members of a monitor group can include services, such as the GSM service, for a group of accounts. When the member is a service, BRM monitors charges for that service only. For example, a company that pays all GSM services for its employees could create a monitor group that includes each employee's GSM service but excludes any other services owned by the employee accounts.

Accounts and services can belong to one or more monitor groups. BRM tracks the monitor groups to which an account or service belongs and then accesses this information during the rating process to determine where to apply balance impacts. For more information, see "About Synchronizing Monitor Data between Real-Time and Pipeline Batch Rating".

About Monitor Group Types

Monitor group types specify the type of members a monitor group contains. For example, group membership can be restricted to nonpaying child accounts and their services, or it can include both paying and nonpaying child accounts and their services.

Table 10–1 shows the monitor group types supported by BRM:

Table 10–1 Supported Monitor Types

Monitor Type	Description
Hierarchy	Includes paying and nonpaying child accounts and their services.
Paying Responsibility	Includes nonpaying child accounts and their services.
Service Level	Includes a member service in a subscription group.

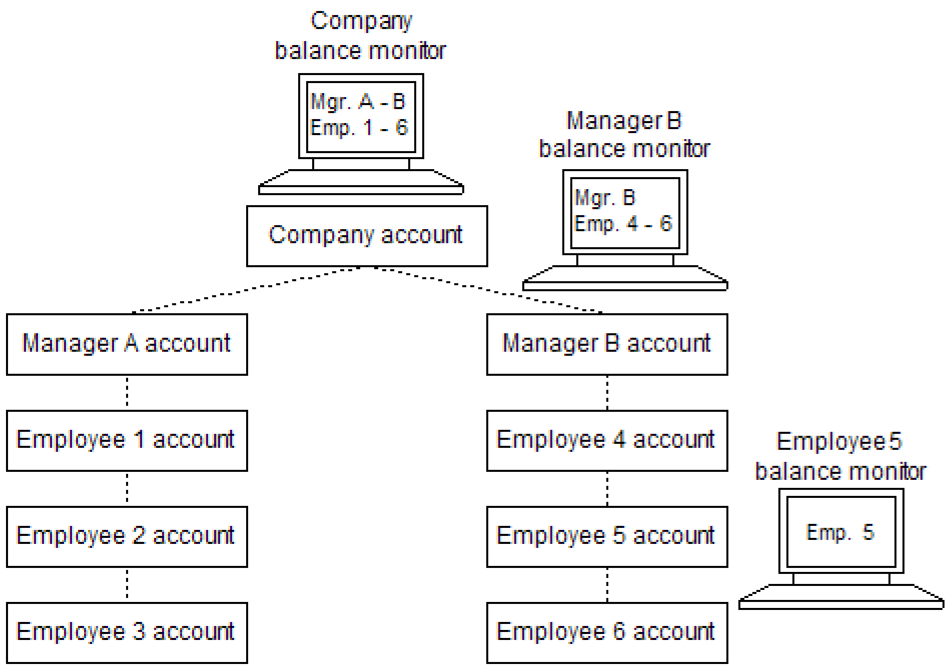
When a customer attempts to add a member to a monitor group, BRM validates the member against rules you specify for each monitor group type. The default implementation contains no validation rules for these monitor group types, but you can create custom group types or create your own rules by customizing the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode. For information, see "Validating the Members of a Balance Monitor Group".

Sample Monitor Groups

Figure 10–1 shows how you can create three different monitor groups in one account hierarchy. In this example, the corporation pays all account charges accumulated by its employees.

- *The company* creates a monitor group to track the balances of its employees. In this case, the owner of the monitor group is the company and the members are all employee accounts.
- *Manager B* creates a separate monitor group to track the balance for his department. In this case, the owner of the monitor group is manager B and the members include the accounts of manager B and all of his direct reports.
- *Employee 5* creates a monitor group to track his own account balance. In this case, the employee is both the owner and the sole member of the monitor group.

Figure 10–1 Monitoring Groups in a Hierarchy



About the Balances of a Monitor Group

The balance for a monitor group is a running total that includes all balance impacts generated by its members. BRM updates the balance when a group member generates a usage event or when you create or modify a monitor group.

BRM stores the total balance for a particular monitor group in **/balance_group/monitor** objects in the database. For information, see "Creating /balance_group/monitor Objects".

Balance Impacts Included in a Monitored Balance

Monitored balance totals include the following:

- **All standard Accounts Receivable (A/R) impacts**, such as these:
  - Discounts
  - Account-level adjustments and disputes
  - Payments
  - Refunds
  - Write-offs

  For more information, see "About Accounts Receivable".
- **Taxes**. Monitored balance totals include taxes that are applied during the real-time rating process only. Any taxation that is deferred to the billing process is not included in the balance total.
- **Sponsored or charge sharing charges**. Monitored balance totals include charges that apply to the sponsor's account only.

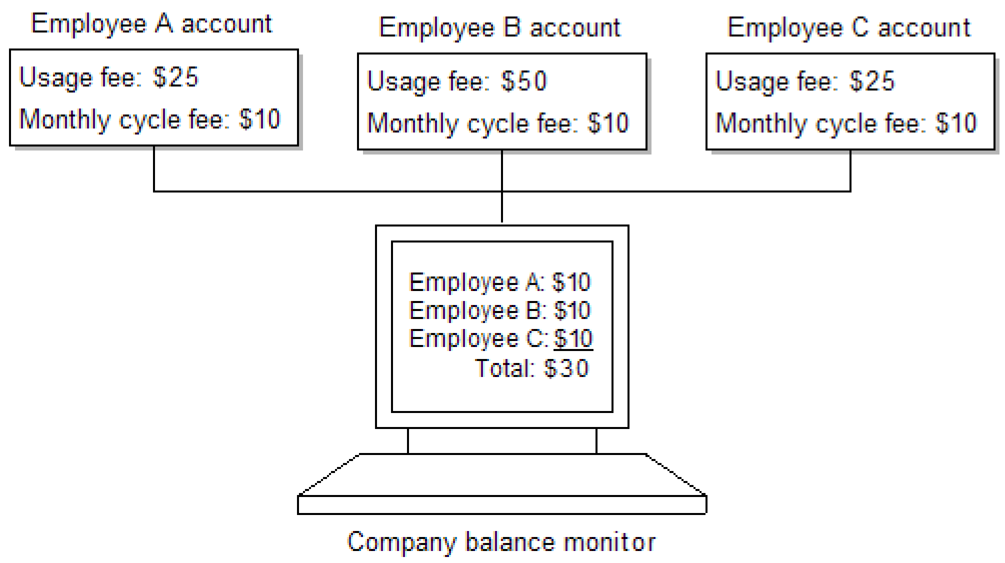
  See "About Monitoring Charge Sharing and Sponsored Account Balances".

About Monitoring Charge Sharing and Sponsored Account Balances

When an account belongs to a charge sharing or sponsor group, its balances may be paid by more than one account. For example, an employee may pay his own wireless usage fees but have his monthly subscription fees paid by his company. When this occurs, each balance monitor tracks only the charges for which it is responsible. In this example, the company's balance monitor tracks the subscription fee, and the employee's balance monitor tracks all usage charges.

For example, assume a company pays only for its employees' monthly subscription fees. If employees A, B, and C are each charged $10 in subscription fees but accrue $100 in usage fees, the company's balance monitor includes only $30 in subscription fees as shown in Figure 10–2.

Figure 10–2 Balance Monitor with Usage and Cycle Fee



When Balance Impacts Are Added to a Monitored Balance

BRM updates the balance of a monitor group at the following times:

- **When a balance monitor is created**. When you first create a balance monitor and run the **pin_monitor_balance** utility, BRM automatically calculates the group balance. BRM retrieves the current balance of each member and adds it to the group balance.

  For more information, see "About Creating and Maintaining Balance Monitors".

- **When members are added to or removed from a monitor group**. When a member is added to a monitor group, BRM automatically adds the member's current balance to the group balance when **pin_monitor_balance** is run. Likewise, when a member is removed from a monitor group and **pin_monitor_balance** is run, BRM automatically decreases the group balance.

  For more information, see "About Creating and Maintaining Balance Monitors".

- **As a result of pipeline rating**. When rating events, the pipeline rating module calculates any monitor balance impacts immediately. However, the impacts are not added to the group balance until you load the data into the BRM database and run the balance monitor utility. Therefore, there is a small lag between the time an event occurs and the time it impacts the group balance.

  For more information, see "About Balance Monitoring and Pipeline Rating".

- **As a result of real-time rating**. When rating events, BRM calculates any monitor balance impacts immediately. However, the impacts are not added to the group balance until you run the balance monitor utility. Therefore, there is a small lag between the time an event occurs and the time it impacts the group balance.

  For more information, see "About Balance Monitoring and Real-Time Rating".

Alerting Customers When Monitored Balances Cross Limits or Thresholds

To be alerted when the balance of their monitor group is approaching predefined values, customers must set up thresholds and credit limits. BRM then tracks the group balances and generates notification events when the balances cross above or below the threshold value or credit limit.

BRM checks whether a group balance has crossed a threshold or a credit limit whenever the following occur:

- The balance of a monitor group is updated
- Monitor group owners add or change limit and threshold values

For information about handling this in Customer Center, see the Customer Center Help.

About Setting Limits and Thresholds

To be alerted when a monitored balance reaches a certain value, customers must define the following for each balance monitor:

- **Credit floor**. The credit floor specifies the starting point for a threshold value. The default credit floor is NULL. You must set the credit floor to a non-NULL value.
- **Credit limit**. The credit limit specifies the ending point for a threshold value. BRM uses the credit limit to calculate the threshold value, which is a percentage of the credit limit. BRM generates a notification event when the group balance crosses above or below the credit limit.

  Note: *Balance monitor credit limits* are separate from *account credit limits*. Account credit limits specify the maximum balance for an individual account and affect the way BRM handles authorizations and rating. Monitor credit limits are used only to calculate threshold values and do not affect authorization and rating.

- **Thresholds**. Thresholds specify the balance totals that trigger an alert to CSRs and monitor group owners. Owners specify thresholds in the following ways:
  - As a percentage of the credit limit in 5% increments (75%, 80%, 85%, and so on). To be notified when the credit limit is reached, select the 100% threshold setting.
  - As a fixed value, such as $90 or 50 minutes.

  Important: You must set a value for each of these parameters. BRM cannot send notification events when any of the following are true: the credit floor is NULL, the credit limit is infinite or undefined, or the threshold is undefined.

For example, a customer might set a monitor group’s credit floor to $0, credit limit to $100, and thresholds to 70%, 90%, and 100%. This causes BRM to generate notification events whenever the monitored balance crosses above or below $70, $90, and $100.

You define a balance monitor's credit floor, credit limit, and threshold values when you create the balance monitor in Customer Center or your custom client application. BRM then stores the values in a **/config/credit_profile** object in the BRM database.

About Using Event Notification to Alert Customers

BRM uses event notification to alert monitor owners when balances cross a threshold or credit limit. When a monitored balance crosses a threshold or limit, the following occurs:

1. BRM generates one of these notification events:
   - **/event/notification/threshold**— Indicates that the balance crossed *above* a threshold value or credit limit.
   - **/event/notification/threshold_below**— Indicates that the balance crossed *below* a threshold value or credit limit.

     For information about these events, see "Event Notification Definitions" in *BRM Developer's Reference*.
2. BRM calls the opcode associated with the event in your system's event notification list. See "About the Event Notification List" in *BRM Developer's Guide*.
3. The opcode processes the event data and passes information to your custom client application.

To enable this, you must configure the event notification feature, the opcode (if it is a policy opcode), and your custom client application to process these events and alert balance monitor owners. For more information, see "Configuring Event Notification for Balance Monitoring".

About Notification Events for Balance Monitoring

BRM generates notification events whenever a monitored balance crosses over or under a credit limit or threshold. This enables you to make BRM perform different actions depending on the event type. For example, you can customize BRM to send an email to both you and the group owner when the group balance crosses above a threshold, but to send an email only to the group owner when the balance falls below a threshold.

About the Number of Notification Events

BRM generates one notification event for each monitored balance that crosses a threshold or credit limit. When one event causes a balance to cross multiple thresholds, BRM generates only one event that lists all settings that were breached.

For example, member A belongs to balance monitors 1 and 2. When member A generates a usage event that crosses the 50% and 55% thresholds for monitor 1, and the 40% threshold for monitor 2, BRM generates only two events:

- The first notification event specifies that the balance of monitor 1 surpassed both the 50% and 55% thresholds.
- The second notification event specifies that the balance of monitor 2 surpassed the 40% threshold.

Information Included in Balance Monitor Notification Events

Each balance monitor notification event includes the following information:

- **Target balance monitor**.

- **Resource ID**.
- **Alert type**: Credit limit or threshold breach.
- **Reason for the breach**:
  - **Upward breach**. The monitored balance crossed above a threshold value.
  - **Downward breach**. The monitored balance crossed below a threshold value, for example, when the customer makes a payment.
  - **Upward reset**. The owner of the balance monitor increased a threshold value, causing the balance to cross below a threshold. For example, this occurs when the current balance is $50 and you change the value of threshold 1 from $40 to $60. In this case, the $50 balance crosses below the new setting for threshold 1.
  - **Downward reset**. The owner of the balance monitor decreased a threshold value, causing the balance to pass above a threshold. For example, this occurs when the current balance is $40 and you change the value of threshold 1 from $50 to $25. In this case, the $40 balance passes above the new setting for threshold 1.
- **Monitor type**: Hierarchy credit exposure, paying responsibility credit exposure, or service level.
- **Source of the breach**.

  Note: BRM does not provide a source ID when the alert is due to a threshold reset or group member addition.

- **List of thresholds or credit limits that were breached**. For example, 50% and 70%. For credit limit breaches, this field is set to 100%.

For more information about these events, see "Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications".

Providing Real-Time Access to the Balances of Monitor Groups

Balance monitor owners can access the balance for their monitor groups in real-time by using a Web interface, such as a Web-enabled phone or computer. When monitor owners log in, BRM collects their account information and optionally a date range and then retrieves the following for each balance monitor:

- Beginning balance of the monitor group on the start date
- Ending balance of the monitor group on the end date

To provide your customers and CSRs with access to real-time monitor balances, you must customize your client interface to use the BRM API. For information, see "Displaying Balance Monitor Information in Client Applications".

About Creating and Maintaining Balance Monitors

You create and modify balance monitors by using a third-party customer relationship management (CRM) application. To implement balance monitoring in your CRM application, you must configure it to accept the following information for each balance monitor and pass it to the BRM API:

- Balance monitor name

- Balance monitor owner
- List of group members
- The monitor group type: hierarchy, paying responsibility, or service level
- Credit limit (optional)
- Credit floor (optional)
- Threshold settings (optional)

Balance Monitor Creation Process Overview

The process that BRM uses to create or modify a balance monitor depends on whether you implement balance monitoring in your system with or without Automated Monitor Setup (AMS).

- **With AMS**, you can automate many of the balance monitor processes. AMS automatically adds and removes members from the balance monitor for you.
- **Without AMS**, you add and remove members manually. You must use this method to create balance monitors for custom monitor types.

Important: The Customer Center balance monitoring functionality requires AMS and does not support adding and removing members manually.

BRM creates balance monitors as follows:

1. Takes as input all information about the balance monitor, such as the monitor owner and threshold settings.
2. Creates a bucket to store the balance of the monitor group.
3. Adds members to the balance monitor. BRM determines whether the following is true:
   - AMS is enabled.
   - The monitor type is hierarchy, paying responsibility, or service level.

   If both are true, BRM uses AMS to add members. BRM uses the monitor owner and monitor type to find all appropriate child accounts and services, and adds them to the monitor. See "About Using AMS to Manage Balance Monitors Automatically".

   If either one is false, BRM adds members without AMS. BRM takes as input the list of members and validates them against custom criteria. All members that pass validation are added to the balance monitor. See "Managing Balance Monitors without AMS".
4. Retrieves the current balance of all members and adds it to the group balance.

About Using AMS to Manage Balance Monitors Automatically

You configure BRM to automatically add and remove members from balance monitors, based on the monitor group type, by using Automated Monitor Setup (AMS). AMS is a group of opcodes that automate many of the balance monitor processes, making balance monitoring easier to set up and use.

AMS creates and updates balance monitors by following the organizational hierarchy of accounts and services. This synchronizes the membership between a hierarchy and a monitor group.

Important: AMS works only for the three default monitor group types: hierarchy, paying responsibility, and service level. For custom group types, you must create balance monitors without AMS. See "Managing Balance Monitors without AMS".

AMS automatically adds and removes members from balance monitors when the following occur:

- **A balance monitor is created**. AMS takes as input the parent (owner) account or service and the monitor type, and then searches through the parent's lineage to find accounts and services that match the criteria for the monitor type. All subordinate accounts and services that meet the criteria are added as members.

  For example, when users create a paying responsibility-type monitor, AMS adds to the monitor the parent account and its services and all subordinate nonpaying child accounts and their services.

- **An account hierarchy or subscription group is changed**. AMS automatically adds or removes members from any associated balance monitors, based on the monitor type. For example, when you add a nonpaying child account to a hierarchy, AMS automatically:

  – Finds all parent accounts at a higher level in the hierarchy than the specified child account.

  – Finds all hierarchy and paying responsibility balance monitors associated with each parent account.

  – Adds the child account and its services to each balance monitor.

When a balance monitor is created or a hierarchy or subscription group changes, the following occurs:

1. BRM generates an AMS notification event.
2. The BRM event notification feature calls one of the AMS opcodes.
3. The AMS opcodes add or remove members from the appropriate monitor group and update the monitored balance.

Managing Balance Monitors without AMS

When creating or updating balance monitors without AMS, users are responsible for adding and removing members. BRM does not verify that balance monitors include all appropriate child accounts and services in a hierarchy or subscription group nor does it check whether a hierarchy or subscription group changes over time.

Users must add or remove members when the following occur:

- **A balance monitor is created**. Users specify the individual accounts and services to add to the balance monitor.
- **An account hierarchy or subscription group is changed**. Users must manually update any balance monitors associated with the hierarchy or subscription group. When an account is added to a hierarchy, users must manually add the account to all balance monitors associated with the group. Likewise, when an account is

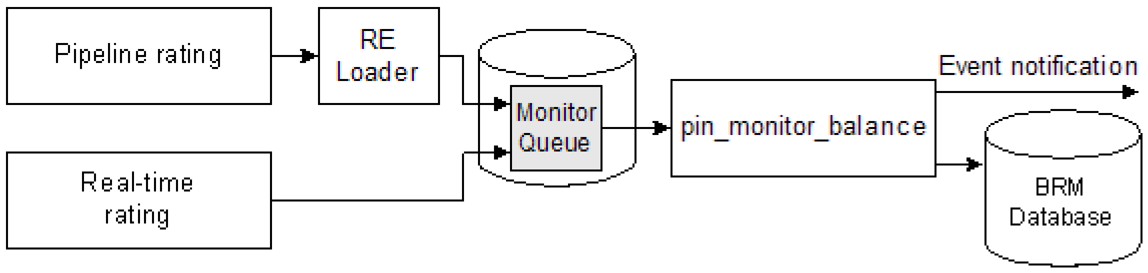
removed from a hierarchy, users must manually remove the account from any associated monitors.

To manage balance monitors without AMS, see "Using the Balance Monitor API to Create and Maintain Balance Monitors".

Balance Monitoring Process Overview

Figure 10–3 shows the balance monitoring process.

Figure 10–3 How BRM Processes Balance Monitoring Data



BRM processes balance monitoring data as follows:

1. BRM rates events and collects any monitored balance impact data.
   - For real-time rated events, see "About Balance Monitoring and Real-Time Rating".
   - For pipeline-rated events, see "About Balance Monitoring and Pipeline Rating".
2. BRM publishes monitored balance impacts to the monitor queue.

   See "Understanding the Monitor Queue".
3. The **pin_monitor_balance** utility retrieves data from the monitor queue and calls the PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode to perform the following:
   - Add balance impacts to the appropriate monitored balance in the BRM database. See "Using pin_monitor_balance to Update Monitored Balances".
   - Determine whether the monitored balance crossed any thresholds and, if it did, generate a notification event. See "About Using Event Notification to Alert Customers".

About Balance Monitoring and Real-Time Rating

Real-time rating processes monitored balances as follows:

1. Rates the event and applies any discounts.
2. Determines whether the event owner belongs to any monitor groups.
3. Determines whether balance monitoring is enabled.
4. Generates monitor balance impacts for each monitor group listed in the **/ordered_ balgrp** object. The monitor balance impact includes the following data:
   - Event owner
   - Amount

- Resource ID
- Balance monitor (POID of the **/balance_group/monitor** object)

5. Publishes the monitor balance impacts to the monitor queue.

   See "Understanding the Monitor Queue".

About Balance Monitoring and Pipeline Rating

Pipeline Manager processes monitored balances as follows:

1. Pipeline modules rate the event and apply any discounts.
2. The FCT_Account module determines whether the event owner belongs to any monitor groups and, if it does, enriches the MONITOR_LIST section of the EDR container with information about each monitor group.

   For information, see "FCT_Account" in *BRM Configuring Pipeline Rating and Discounting*.

   > **Note:** DAT_AccountBatch stores in-memory the list of monitor groups to which an account or service belongs. The module retrieves this information from the BRM database when you first start Pipeline Manager and when you create, modify, or delete a monitor group. For more information, see "About Synchronizing Monitor Data between Real-Time and Pipeline Batch Rating".

3. The FCT_BillingRecord module generates a monitor packet for each monitor group.

   - When a balance packet impacts the event balance group, FCT_BillingRecord creates a MONITOR_PACKET for each monitor group in the monitor list.
   - When a balance packet does not impact the event balance group (for example, the discount share or charge share event balance groups), FCT_BillingRecord calls DAT_AccountBatch to retrieve the monitor group list for the discount share or charge share owner's balance group.

   The MONITOR_PACKET contains an account POID of the balance monitor, the balance group ID from the monitor group, resource ID, amount, sub-balance impact, and the sub-balance block from the balance packet.

   For information, see "FCT_BillingRecord" in *BRM Configuring Pipeline Rating and Discounting*.

4. The pipeline generates an output file that includes any monitor balance impacts.
5. Rated Event (RE) Loader retrieves the pipeline output file and publishes any monitored balance impact data to the monitor queue.

   See "Understanding the Monitor Queue".

Understanding the Monitor Queue

The monitor queue (**/monitor_queue**) holds all monitor balance impacts generated by real-time rating and pipeline batch rating. The **pin_monitor_balance** utility retrieves this information and updates the balances of the associated monitor groups.

For performance reasons, the **pin_monitor_balance** utility does not delete **/monitor_queue** objects after they are retrieved from the BRM database. Instead, the utility flags the **/monitor_queue** objects as "processed" to prevent the objects from being consumed again.

BRM tracks the status of **/monitor_queue** objects by using the PIN_FLD_STATUS field:

- When the field is set to **0**, the object has not been processed by the **pin_monitor_balance** utility.
- When the field is set to a **nonzero value**, the object has been processed by the utility.

Using pin_monitor_balance to Update Monitored Balances

You use the **pin_monitor_balance** utility to apply monitor balance impacts to a monitored balance. This utility is a multithreaded application (MTA) that retrieves a list of all **/monitor_queue** objects in the BRM database and then spawns child threads to process each object individually.

You can configure the utility to run with one thread or multiple threads:

- One thread enables the utility to process events in the order that they occurred but decreases processing performance.
- Multiple threads increase performance but may cause the utility to process events out of order. The default is 5 threads.

To configure the number of threads, see "Configuring pin_monitor_balance to Process Events in the Order Created".

You can run **pin_monitor_balance** manually or configure a scheduler, such as **cron**, to run it at predefined intervals. For more information, see "Running pin_monitor_balance to Update Monitored Balances".

The **pin_monitor_balance** utility performs the following tasks:

1. Retrieves all unprocessed **/monitor_queue** objects from the BRM database (that is, all objects with the PIN_FLD_STATUS field set to **0**).
2. Retrieves the **/event** object associated with the **/monitor_queue** object to obtain the following:
   - Monitor balance impacts
   - Sub-balance impacts
   - Target balance monitors (**/balance_group/monitor** objects)
   - Event owner
3. Calls the PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode, which performs the following:
   - Updates the monitored balance (**/balance_group/monitor**).
   - Determines whether any balance monitor thresholds have been breached by comparing the monitored balance to the threshold and credit limit values stored in the **/config/credit_profile** object. If any thresholds or limits were breached, the opcode generates a notification event. See "About Using Event Notification to Alert Customers".
   - Sets the **/monitor_queue** object's PIN_FLD_STATUS field to a nonzero value to prevent it from being processed again by the utility.

About Synchronizing Monitor Data between Real-Time and Pipeline Batch Rating

Pipeline batch rating and real-time rating determine where to apply balance monitor impacts by accessing each member's list of monitor groups. For real-time rating, this information is stored in each member's **/ordered_balgrp** object in the BRM database. For pipeline batch rating, this information is stored in memory by the DAT_AccountBatch module.

When you create, modify, or delete a balance monitor, BRM updates each member's **/ordered_balgrp** object and then uses Account Synchronization to send the updated information to Pipeline Manager, as follows:

1. BRM updates the member's **/ordered_balgrp** object and generates one of the business events listed in Table 10–2:

   Table 10–2 Monitor-Related Actions

Action	Event
Create a new monitor group	**/event/group/sharing/monitor/create**
Modify a monitor group by adding or removing members	**/event/group/sharing/monitor/modify**
Delete a monitor group	**/event/group/sharing/monitor/delete**

   For more information on these storable objects, see *BRM Storable Class Reference*.

2. Account Synchronization processes these business events and publishes them to the Account Synchronization queue.

   For more information, see "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.

3. Pipeline Manager retrieves the data from the queue and updates its internal memory:

   - The DAT_Listener module retrieves the data from the queue and passes it to DAT_AccountBatch.
   - DAT_AccountBatch refreshes the monitor group list for all accounts.

   For more information, see "Pipeline Manager Reference" in *BRM Configuring Pipeline Rating and Discounting*.

Rerating Events When Members Are Added to Balance Monitors

The account synchronization process produces a small lag between the time a member's **/ordered_balgrp** object is updated in the BRM database and the time DAT_AccountBatch is updated. During this lag time, the following can occur:

- Pipeline rating generates balance impacts for new balance monitor members, but does not generate monitor balance impacts.
- Pipeline-rated events that have not been loaded into the database do not include monitor balance impacts for the new members.

This causes a discrepancy between the amount added to the member's account balance and the amount added to the balance monitor. To capture any missing monitor balance impacts, BRM automatically rerates events that were generated by the new balance monitor members.

When members are added to a balance monitor, the following occurs:

1. Balance monitoring generates a notification event.
2. Pipeline Manager suspends any incoming events that are generated by the new balance monitor members.
3. Rated Event (RE) Loader loads all pipeline output files into the database.
4. The **pin_rerate** utility rerates all events that were generated by the new balance monitor members. The utility does the following:
   a. Flushes all monitor balance impacts from the **/monitor_queue**.
   b. Backs out all events that were rated during the lag time.
   c. Rerates the events and generates monitor balance impacts.
   d. Applies monitor balance impacts to the appropriate balance monitor.

   For more information about **pin_rerate**, see "About Comprehensive Rerating Using pin_rerate" in *BRM Setting Up Pricing and Rating*.
5. Pipeline Manager resumes processing EDRs generated by the new monitor members and recycles all EDRs suspended in step 2.

Important: You must ensure that RE Loader has sufficient time to load all pipeline-rated events into the database before rerating begins. To do this, configure **pin_rerate** to wait a specified amount of time before starting the rerating process. See "Specifying a Wait Time before Rerating Events".

Configuring Your BRM System for Balance Monitoring

To configure your system for balance monitoring, perform the following tasks:

1. Enable balance monitoring in BRM.

   See "Enabling Balance Monitoring in BRM".
2. Enable AMS in BRM.

   See "Enabling AMS in BRM".
3. Enable balance monitoring in Pipeline Manager.

   See "Enabling Balance Monitoring in Pipeline Manager".
4. Configure BRM to perform appropriate follow-up operations when limits and thresholds are crossed.

   See "Configuring Event Notification for Balance Monitoring".
5. Configure **pin_rerate** to wait a specified amount of time before starting the rerating process.

   See "Specifying a Wait Time before Rerating Events".
6. (Optional) Configure **pin_monitor_balance** to process monitor balance impacts in the order that they were created.

   See "Configuring pin_monitor_balance to Process Events in the Order Created".

Enabling Balance Monitoring in BRM

By default, balance monitoring is disabled in BRM. You can enable this feature by modifying a field in the **multi-bal** instance of the **/config/business_params** object.

When balance monitoring is enabled, BRM monitors currency resources.

You modify the **/config/business_params** object by using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

To enable balance monitoring:

1. Use the following command to create an editable XML file from the **multi-bal** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

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

2. Search the XML file for following line:

   ```
   <BalanceMonitoring>disabled</BalanceMonitoring>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **multi-bal** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM balance monitoring configuration.

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

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

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

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

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using testnap" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

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

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Enabling AMS in BRM

By default, AMS is disabled in BRM. You can enable this feature to automate many of the balance monitoring processes by modifying a field in the **subscription** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

To enable AMS:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

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

2. Search the XML file for following line:

   ```
   <AutomatedMonitorSetup>disabled</AutomatedMonitorSetup>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **subscription** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM balance monitoring configuration.

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

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

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using testnap" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects by Using Object Browser" in *BRM Developer's Guide*.

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

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Enabling Balance Monitoring in Pipeline Manager

Pipeline Manager determines whether balance monitoring is enabled by using the **BalanceMonitoring** entry from the **multi-bal** instance of the **/config/business_ params** object. See "Enabling Balance Monitoring in BRM".

You must configure Pipeline Manager to use business parameter settings from the BRM database by performing the following:

- Configuring the DAT_PortalConfig module in your registry file (see "DAT_ PortalConfig" in *BRM Configuring Pipeline Rating and Discounting*). This module must be listed before all other data modules in the registry file.
- Connecting the DAT_AccountBatch module to DAT_PortalConfig by using the **PortalConfigDataModule** registry entry (see "DAT_AccountBatch" in *BRM Configuring Pipeline Rating and Discounting*).

For more information, see "Using Business Parameter Settings from the BRM Database" in *BRM System Administrator's Guide*.

Important: You must enable balance monitoring in Pipeline Manager to have notification events generated during the batch rating process.

Configuring Event Notification for Balance Monitoring

When the balance of a monitor group crosses a threshold or a credit limit, BRM generates the following notification events. You must configure the BRM event notification feature to call opcodes that perform the appropriate follow-up operations when these events are generated.

- **/event/notification/threshold**: This event indicates that the group's balance reached or exceeded a threshold value; for example, when a purchase causes the balance to go over a 70% threshold.

  By default, when this event occurs, the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode is called. You can add custom code to this policy opcode that processes upward threshold breaches, or you can enable this event to trigger a call to a balance monitoring opcode that processes upward threshold breaches (see "Editing the Event Notification List" in *BRM Developer's Guide*).
- **/event/notification/threshold_below**: This event indicates that the group's balance passed below a threshold value; for example, when the customer pays a bill. This event is generated only when the group's balance is less than the threshold value.

  By default, when this event occurs, the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode is called. To enable this event to trigger a call to a balance monitoring opcode that processes downward threshold breaches, see "Editing the Event Notification List" in *BRM Developer's Guide*.

For more information about these events, see "About Notification Events for Balance Monitoring".

To configure the event notification feature for balance monitoring:

1. If your system has multiple configuration files for event notification, merge them.

   See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Add information to your final event notification list about the appropriate opcodes to call when the **/event/notification/threshold** and **/event/notification/threshold_ below** events occur.

See "Editing the Event Notification List" in *BRM Developer's Guide*.

Note:

- The default *BRM_Home***/sys/data/config/pin_notify** file includes the following line, which means that the PCM_OP_ACT_POL_EVENT_NOTIFY (opcode number 301) policy opcode is called whenever an **/event/notification/threshold** event occurs:

  ```
  301    0    /event/notification/threshold
  ```

- The number of each balance monitoring opcode is in the balance monitor opcode header file, *BRM_Home***/include/ops/monitor.h**.

---

3. If you enabled AMS, ensure that the file includes the following information from the *BRM_Home***/sys/data/config/pin_notify** file:

```
# Automated Monitor Setup related event notification
# Uncomment these lines when AMS feature is enabled
7853    0       /event/group/sharing/monitor/create
7854    0       /event/notification/service/pre_purchase
7855    0       /event/notification/service/pre_purchase
7856    0       /event/notification/service/pre_purchase
7857    0       /event/customer/billinfo/modify
7855    0       /event/customer/billinfo/modify
7857    0       /event/group/member
7854    0       /event/group/member
7857    0       /event/notification/bal_grp/modify
7855    0       /event/notification/bal_grp/modify
7854    0       /event/audit/subscription/transfer
3787    0       /event/billing/monitor/update
```

Important: If the lines associating opcode numbers with events are commented out, uncomment them.

---

4. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.

   See "Editing the Event Notification List" in *BRM Developer's Guide*.

5. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger.

   See "Triggering Custom Operations" in *BRM Developer's Guide*.

6. Load your final event notification list into the BRM database.

   See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Specifying a Wait Time before Rerating Events

You must configure **pin_rerate** to wait a specified amount of time before rerating events. For more information, see "Rerating Events When Members Are Added to Balance Monitors".

To configure a wait time:

1. Open the **pin_rerate** configuration file (*BRM_Home*/**apps/pin_rerate/pin.conf**) in a text editor.
2. Add the following entry and specify the delay time in seconds. Make sure to allow enough time for RE Loader to load all of the pipeline-rated data into the database.

   Note: The default is **0**.

   ```
   - pin_rerate        delay        300
   ```

3. Save and close the file.

Configuring pin_monitor_balance to Process Events in the Order Created

To process monitor balance impacts in chronological order, configure the **pin_monitor_balance** utility to run with only one thread. For more information, see "Using pin_monitor_balance to Update Monitored Balances".

To process monitor balance impacts in chronological order, perform the following tasks:

1. Open the **pin_monitor_balance** configuration file (*BRM_Home*/**apps/pin_monitor/pin.conf**) in a text editor.
2. Change the following entry to **1**.

   ```
   - pin_mta   children   1
   ```

3. Save and close the file.

Running pin_monitor_balance to Update Monitored Balances

To update monitored balances, perform the following:

1. Go to the *BRM_Home*/**apps/pin_monitor** directory.
2. Run the **pin_monitor_balance** utility:

   Note: To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

   ```
   pin_monitor_balance [-d] [-v]
   ```

   For more information, see **pin_monitor_balance**.

Specifying Whether Item Transfers Affect Balance Monitors

BRM updates balance monitor totals when you perform item adjustments, settlements, and disputes. By default, BRM also updates balance monitor totals when you perform item transfers. This can cause BRM to apply double the dispute, adjustment, or settlement amount to a balance monitor total when correcting misapplied charges. For example, when a customer disputes a misapplied charge, BRM reduces the balance monitor total when the charge is disputed and then again when the charge is transferred from the customer's account to another account.

You can configure BRM to not apply balance impacts from item transfers to balance monitor totals by passing the optional PIN_FLD_APPLY_MONITOR input flist field to the PCM_OP_BILL_ITEM_TRANSFER opcode:

- When PIN_FLD_APPLY_MONITOR is set to **0**, balance impacts from item transfers *are not* added to balance monitor totals.
- When PIN_FLD_APPLY_MONITOR is set to **1** or not passed, balance impacts from item transfers *are* added to balance monitor totals. This is the default.

For more information about adjustments, disputes, and settlements, see "Configuring Adjustments, Disputes, and Settlements".

Implementing Balance Monitoring in Custom Client Applications

Before reading the following, you should be familiar with these concepts in BRM:

- **Balance monitoring.** For more information, see "About Balance Monitoring".
- **Balance groups.** For more information, see "About Balance Groups".
- **Charge sharing.** For more information, see "About Charge Sharing Groups".
- **Discount sharing.** For more information, see "About Discount Sharing Groups".
- **Rating.** For more information, see "How Rating Works" in *BRM Setting Up Pricing and Rating*.

For information about the balance monitoring interface in Customer Center, see the Customer Center Help.

About Implementing Balance Monitoring in Client Applications

To monitor balances, your Customer Service Representative (CSR) must be able to perform the following tasks:

- Specify the group of account and service balances to monitor.
- Update the group by adding or deleting members.
- View the balance monitors owned by an account.
- View the current balance of a balance monitor.
- Receive alerts when a balance monitor’s credit limit or threshold is crossed.

Note: For information about the balance monitoring interface in Customer Center, see the Customer Center Help.

To implement balance monitoring in your client application, add the following functionality to your application by using the BRM opcodes:

- Create or update balance monitors.

  See "Creating and Maintaining Balance Monitor Objects".
- Display the balances for a monitor group or a list of monitors owned by an account or service.

  See "Displaying Balance Monitor Information in Client Applications".
- Send notifications when a credit limit or threshold is crossed.

See "Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications".

Your user interface must be designed to collect the information needed for creating or updating the relevant objects and pass the appropriate fields in the input flist to the opcodes.

Creating and Maintaining Balance Monitor Objects

To create or update a balance monitor, you must store the following balance monitor data in the BRM database:

1. The total rolled-up balance in a **/balance_group/monitor** object.

   You create only one such object for each balance monitor.

2. The list of monitor members in a **/group/sharing/monitor** object.

   You create only one such object for each balance monitor.

3. The list of monitor groups that each member belongs to in **/ordered_balgrp** objects.

   You must create or update one such object for each account and service in a balance monitor.

The method you use to create and update these objects depends on whether you are using Automated Monitor Setup (AMS), which is a group of opcodes that automate how BRM creates and maintains balance monitors.

- To create or update balance monitors by using AMS, see "Using AMS to Create and Maintain Balance Monitors Automatically".
- To create or update balance monitors without AMS, see "Using the Balance Monitor API to Create and Maintain Balance Monitors".

Using AMS to Create and Maintain Balance Monitors Automatically

For more information about AMS, see "About Using AMS to Manage Balance Monitors Automatically".

For information about using the balance monitor interface in Customer Center, see the Customer Center Help.

To use AMS to create and update balance monitors automatically, you must configure your custom client application to call the following opcodes:

1. PCM_OP_CUST_SET_BAL_GRP: This opcode creates and updates the **/balance_group/monitor** object.

   - To create the object, see "Creating /balance_group/monitor Objects".
   - To modify the object, see "Modifying /balance_group/monitor Objects".

2. PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE: This opcode creates the **/group/sharing/monitor** object.

   You must pass the monitor group type in the input flist. To create and maintain these objects, see "Creating, Modifying, or Deleting /group/sharing/monitor Objects".

When a **/group/sharing/monitor** object is created or modified, BRM uses the event notification feature to trigger calls to the appropriate AMS opcodes. These opcodes automatically add and remove members from **/group/sharing/monitor** objects and update each member's **/ordered_balgrp** object based on the monitor type. See "Adding and Removing Balance Monitor Members Automatically".

Important: For subscription groups only, create a separate **/ordered_balgrp** object for any member service that uses a separate **/billinfo** object than its parent subscription service. You create **/ordered_balgrp** objects manually by using the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See "Adding a Monitor Group to a Member's /ordered_balgrp Object".

Using the Balance Monitor API to Create and Maintain Balance Monitors

For more information, see "Managing Balance Monitors without AMS".

To create or update balance monitors without AMS, you must configure your custom client application to call the following opcodes:

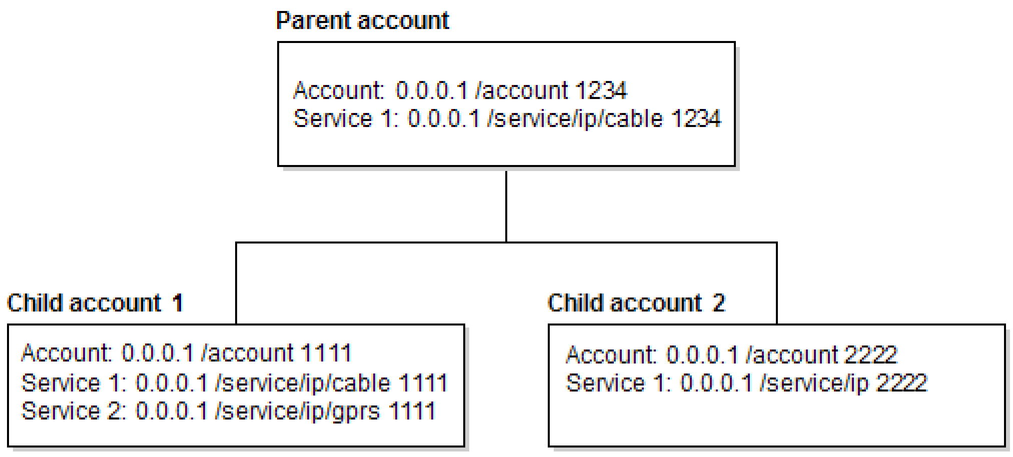
1. PCM_OP_CUST_SET_BAL_GRP: This opcode creates and updates the **/balance_group/monitor** object.
   - To create the object, see "Creating /balance_group/monitor Objects".
   - To modify the object, see "Modifying /balance_group/monitor Objects".
2. PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE: This opcode creates the **/group/sharing/monitor** object.

   You must pass all group members in the input flist.
   - To create, modify, or delete the object, see "Creating, Modifying, or Deleting /group/sharing/monitor Objects".
   - To validate potential members before adding them to the object, see "Validating the Members of a Balance Monitor Group".
3. PCM_OP_SUBSCRIPTION_ORDERED_BALGRP: This opcode creates and updates **/ordered_balgrp** objects.

   You must create or update the **/ordered_balgrp** object associated with each account and service that you want to include in a balance monitor. See "Adding a Monitor Group to a Member's /ordered_balgrp Object".

   For example, assume a balance monitor includes the following account hierarchy shown in Figure 10–4:

Figure 10–4 Example Account Hierarchy with Balance Monitoring


Parent account
Account: 0.0.0.1 /account 1234
Service 1: 0.0.0.1 /service/ip/cable 1234
Child account 1
Account: 0.0.0.1 /account 1111
Service 1: 0.0.0.1 /service/ip/cable 1111
Service 2: 0.0.0.1 /service/ip/gprs 1111
Child account 2
Account: 0.0.0.1 /account 2222
Service 1: 0.0.0.1 /service/ip 2222

To include in the balance monitor the balance of all three accounts and their services, create or update the **/ordered_balgrp** object associated with each of the following:

- **/account 1234**
- **/service/ip/cable 1234**
- **/account 1111**
- **/service/ip/cable 1111**
- **/service/ip/gprs 1111**
- **/account 2222**
- **/service/ip 2222**

Creating /balance_group/monitor Objects

To create a **/balance_group/monitor** object, call the PCM_OP_CUST_SET_BAL_GRP opcode with the following information in the input flist:

- The owner of the balance monitor:
  - When the owner is an account, the POID of the **/account** object
  - When the owner is a service, the POIDs of the **/account** and **/service** objects
- Monitor name
- Credit limit
- Credit floor
- Credit thresholds

Note: You can also pass this information in the PIN_FLD_ACCTINFO array of the input flist to PCM_OP_CUST_COMMIT_CUSTOMER when creating an account. This opcode automatically calls PCM_OP_CUST_SET_BAL_GRP to create a **/balance_group/monitor** object when the relevant information is in the input flist and balance monitoring is enabled. For more information, see "Creating Balance Groups".

Modifying /balance_group/monitor Objects

To modify an existing **/balance_group/monitor** object, call the PCM_OP_CUST_SET_BAL_GRP opcode with the following information in the input flist:

- The owner of the balance monitor:
  - When the owner is an account, the POID of the **/account** object
  - When the owner is a service, the POIDs of the **/account** and **/service** objects
- The balance monitor to modify (POID of the **/balance_group/monitor** object)
- Monitor name
- Credit limit
- Credit floor
- Credit thresholds

Note: You can also pass this information in the PIN_FLD_ACCTINFO array of the input flist to PCM_OP_CUST_UPDATE_CUSTOMER when modifying an account. This opcode automatically calls PCM_OP_CUST_SET_BAL_GRP to modify the **/balance_group/monitor** object when the relevant information is in the input flist and balance monitoring is enabled. For more information, see "Managing Balance Groups with Your Custom Application".

Deleting /balance_group/monitor Objects

You cannot delete **/balance_group/monitor** objects. You can, however, deactivate the monitor group by deleting its associated **/group/sharing/monitor** object. To do so:

1. Run the **pin_monitor_balance** utility to update balances.
2. Delete the associated **group/sharing/monitor** object by calling the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode.

   This opcode removes the reference to the **group/sharing/monitor** object from corresponding **/ordered_balgrp** object. See "Creating, Modifying, or Deleting /group/sharing/monitor Objects".

Creating, Modifying, or Deleting /group/sharing/monitor Objects

To create or modify a **/group/sharing/monitor** object, you must collect the following information in your client application and pass it to the opcodes:

- Owner of the balance monitor:
  - When the owner is an account, the POID of the **/account** object
  - When the owner is a service, the POIDs of the **/account** and **/service** objects
- Monitor group type: hierarchy, paying responsibility, or service level
- List of members you want to add or delete
- Balance monitor associated with the monitor group

You call the following opcodes from your client application:

- To *create* a monitor group, call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode.
- To *modify* a monitor group, call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY opcode.
- To *delete* a monitor group, call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode.

These opcodes perform the following tasks to add, modify, or delete a monitor group object:

1. Verify balance monitoring is enabled.
2. (**Create and modify opcodes only**) Call the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode to validate the members.

   See "Validating the Members of a Balance Monitor Group".
3. Create, modify, or delete the **/group/sharing/monitor** object.

Changing the Owner of a Balance Monitor

To change the owner of a **/group/sharing/monitor** object, call the PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT opcode. For information on the fields you must include in the input flist, see "Changing the Owner of a Resource Sharing Group".

Important: This opcode can be used to change the owner of a balance monitor only if you are performing the change manually. Automated Monitor Setup (AMS) does not support changing owners.

If successful, this opcode returns the POID of the balance monitor that was modified and the event that was generated to record the owner change.

This opcode fails if the new owner that is passed is already a member of the resource sharing group.

To change the owner of a balance monitor, PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - AMS is not enabled. See "Enabling AMS in BRM".
   - The input flist contains the **/balance_group** object and **/account** object for the new balance monitor owner.
   - The **/balance_group** object belongs to the new owning account or service.
2. Sets the **/group/sharing/monitor** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
3. Creates an **/event/group/sharing/monitor/modify** event to record the owner change.

Validating the Members of a Balance Monitor Group

To validate the members of a balance monitor, call the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode. This opcode validates members before they are added or modified based on the monitor type, and returns a list of valid members to the calling opcode.

By default, this opcode validates the hierarchy, paying responsibility, and service level monitor types. You can create any type of monitor and validate it by implementing validation rules in this opcode.

The default implementation of the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS opcode performs the following tasks:

1. Retrieves the monitor type.
2. For each member in the input flist, retrieves the account object.
3. Applies validation rules for the monitor type and returns a list of valid members for the group.

Adding a Monitor Group to a Member's /ordered_balgrp Object

The **/ordered_balgrp** object stores the list of sharing groups to which an account or service belongs. This object controls the order in which credits and discounts are applied and tracks the balance impacts for the balance groups.

For more information on sharing groups, see the Customer Center Help.

Monitor groups are added to the PIN_FLD_ORDERED_BALGROUPS array of the **/ordered_balgrp** object. Unlike the charge sharing and discount sharing groups, the order of the monitor groups does not affect the balances. Therefore, the monitor groups are added to the end of the list.

To add a monitor group to the **/ordered_balgrp** object of an account or service, call the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. This opcode performs the following tasks, depending on the value passed in the PIN_FLD_ACTION field in the input flist:

- Creates or deletes an **/ordered_balgrp** object.
- Modifies an **/ordered_balgrp** object by adding or deleting groups.
- Lists all sharing groups to which the account or service belongs.

This opcode performs the following tasks:

1. Verifies that the input flist has a **/group/sharing/monitor** object in the PIN_FLD_ORDERED_BALGROUPS array and that balance monitoring is enabled.

   Note:

   - When the input flist does not contain a /group/sharing/monitor object, the opcode continues with the tasks for processing the /group/sharing/charges and /group/sharing/discounts objects
   - When balance monitoring is not enabled, the opcode returns an error.

   ---

2. Rearranges the PIN_FLD_ORDERED_BALGROUPS array to place the monitor groups at the end of the sequence.
3. Does one of the following tasks:
   - When the opcode is called to *create* the object, creates the **/ordered_balgrp** object for the specified account or service.
   - When the opcode is called to *modify* the object, modifies the **/ordered_balgrp** object by adding or deleting monitor group objects.
   - When the opcode is called to *delete* the object, deletes the **/ordered_balgrp** object.
4. Retrieves the current balance for the account or service **/balance_group**.
5. Generates an **/event/billing/monitor/update** or **/event/billing/monitor/delete** event.
6. Updates the balance of all associated **/balance_group/monitor** objects.

For more information about PCM_OP_SUBSCRIPTION_ORDERED_BALGRP, see "Managing Ordered Balance Groups".

Adding and Removing Balance Monitor Members Automatically

Use these AMS opcodes to manage balance monitors automatically.

Important: Do not call these opcodes directly. Call these opcodes only through the BRM event notification feature. See "Configuring Event Notification for Balance Monitoring".

- To automatically add members to a balance monitor when it is first created, use PCM_OP_MONITOR_SETUP_MEMBERS.

  See "Adding Members to Newly Created Balance Monitors Automatically".

- To automatically add or remove members from a balance monitor when an account hierarchy or subscription group changes, use these opcodes:
  - PCM_OP_MONITOR_ACCOUNT_HIERARCHY. See "Updating Hierarchy-Type Monitors Automatically".
  - PCM_OP_MONITOR_BILLING_HIERARCHY. See "Updating Paying Responsibility-Type Monitors Automatically".
  - PCM_OP_MONITOR_SERVICE_HIERARCHY. See "Updating Subscription-Type Monitors Automatically".
  - PCM_OP_MONITOR_HIERARCHY_CLEANUP. See "Removing Members from Hierarchy- and Paying Responsibility-Type Monitors".

Adding Members to Newly Created Balance Monitors Automatically

Use the PCM_OP_MONITOR_SETUP_MEMBERS opcode to automatically add members to a balance monitor when it is first created. This opcode is a wrapper opcode that, according to monitor group type, calls other standard MONITOR opcodes to:

- Find and add members to the monitor group (**/group/sharing/monitor** object).
- Update each member's ordered balance group list (**/ordered_balgrp** object).
- Add each member's current balance to the group balance (**/balance_group/monitor** object).

This opcode is triggered by the **/event/group/sharing/monitor/create** event.

Important: Do not call this opcode directly. This opcode should be called through the event notification feature only. See "Configuring Event Notification for Balance Monitoring".

This opcode retrieves the monitor type and monitor owner, and then calls one of these opcodes:

- **For hierarchy group types (H_CE)**: Calls PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS. This opcode adds the following members to the balance monitor:
  - The parent account and its services
  - All *nonpaying* child accounts and their services
  - All *paying* child accounts and their services
- **For paying responsibility group types (PR_CE)**: Calls PCM_OP_MONITOR_PROCESS_BILLING_MONITORS. This opcode adds the following members to the balance monitor:
  - The parent account and its services
  - All *nonpaying* child accounts and their services

- **For service group types (SUB_CE)**: Calls PCM_OP_MONITOR_PROCESS_SERVICE_MONITORS. This opcode adds the following members to the balance monitor:
  - The parent subscription service
  - All member services

For more information about these opcodes, see "Balance Monitoring FM Standard Opcodes" in *BRM Developer's Reference*.

Adding Members to Hierarchy-Type Monitors

Use the PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS opcode to add members to hierarchy-type balance monitors (see *BRM Developer's Reference*). This opcode finds and adds the following members to the balance monitor:

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

This opcode is called directly by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

PCM_OP_MONITOR_PROCESS_HIERARCHY_MONITORS performs the following tasks:

1. Finds all services that are associated with the parent account.
2. Finds in the hierarchy all paying and nonpaying child accounts that belong to the parent account.
3. Finds all services that are associated with each child account.
4. Adds the following members to the balance monitor:
   - The parent account and its services.
   - All child accounts and their services.

   See "Creating, Modifying, or Deleting /group/sharing/monitor Objects".
5. Updates each account's and service's **/ordered_balgrp** object.

   See "Adding a Monitor Group to a Member's /ordered_balgrp Object".

Adding Members to Payment Responsibility-Type Monitors

Use the PCM_OP_MONITOR_PROCESS_BILLING_MONITORS opcode to add members to payment responsibility-type balance monitors. This opcode finds and adds the following members to the balance monitor:

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

This opcode is called directly by the PCM_OP_MONITOR_SETUP_MEMBERS wrapper opcode.

PCM_OP_MONITOR_PROCESS_BILLING_MONITORS performs the following tasks:

1. Retrieves the parent's bill unit (**billinfo**) from either the input flist or searches....
2. Searches for all the **bill_info** objects that have PIN_FLD_PARENT_BILLINFO_OBJ as the current billinfo.

3. Searches for the owning account and all services for each billinfo found.
4. Adds the following members to the balance monitor:
   - The parent account and its services
   - All nonpaying child accounts and their services.

   See "Creating, Modifying, or Deleting /group/sharing/monitor Objects".
5. Updates each account's and service's **/ordered_balgrp** object.

   See "Adding a Monitor Group to a Member's /ordered_balgrp Object".

Updating Hierarchy-Type Monitors Automatically

Use the PCM_OP_MONITOR_ACCOUNT_HIERARCHY opcode to add members to hierarchy-type balance monitors when an account hierarchy changes (see *BRM Developer's Reference*). For example, this opcode adds members to balance monitors when any of the following actions affect an account hierarchy:

- An account is added to the account hierarchy
- A child account adds a service
- The association between a balance group and a bill unit changes
- A line is transferred to an account member

This opcode is triggered by the following events:

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

Important: Do not call this opcode directly. This opcode should be called through the event notification feature only. See "Configuring Event Notification for Balance Monitoring".

PCM_OP_MONITOR_ACCOUNT_HIERARCHY finds all parent accounts at a higher level in the hierarchy than the specified account or service. For each parent account, the opcode performs the following tasks:

1. Finds all hierarchy-type monitors associated with the parent account.
2. Adds the account or service to the hierarchy-type monitor groups (**/group/sharing/monitor** objects).
3. Adds the monitor group to the account's or service's ordered balance group list (**/ordered_balgrp** object).
4. Adds the account or service balance to the monitored balance (**/balance_group/monitor** objects).

Updating Paying Responsibility-Type Monitors Automatically

Use the PCM_OP_MONITOR_BILLING_HIERARCHY opcode to add members to paying responsibility-type monitors when changes occur in an account hierarchy. For example, this opcode adds members to balance monitors when any of the following actions affect an account hierarchy:

- A nonpaying child account is added to the account hierarchy

- A nonpaying child account adds a service
- A child account changes from a paying to a nonpaying payment type
- The association between a balance group and a bill unit changes
- A line is transferred to a nonpaying child account

This opcode is triggered by the following events:

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

Important: Do not call this opcode directly. This opcode should be called through the event notification feature only. See "Configuring Event Notification for Balance Monitoring".

PCM_OP_MONITOR_BILLING_HIERARCHY finds all parent accounts at a higher level in the hierarchy than the specified account or service. For each parent account, the opcode performs the following tasks:

1. Finds all paying responsibility-type monitors associated with the parent account.
2. Adds the account or service to the paying responsibility-type monitor groups (**/group/sharing/monitor** objects).
3. Adds the monitor group to the account's or service's ordered balance group list (**/ordered_balgrp** object).
4. Adds the account or service balance to the balance monitors (**/balance_group/monitor** objects).

Updating Subscription-Type Monitors Automatically

Use the PCM_OP_MONITOR_SERVICE_HIERARCHY opcode to add members to subscription-type monitors when a subscription group changes. For example, when you add a member service to a subscription, this opcode adds the service to any associated balance monitors.

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

Important: Do not call this opcode directly. This opcode should be called through the event notification feature only. See "Configuring Event Notification for Balance Monitoring".

PCM_OP_MONITOR_SERVICE_HIERARCHY finds all parent subscription services at a higher level in the group than the specified service. For each parent subscription service, the opcode performs the following tasks:

1. Finds all subscription-type monitors associated with the parent subscription service.
2. Adds the service to the subscription-type monitor groups (**/group/sharing/monitor** objects).
3. Determines whether the added service is a subscription service:

- If it is, proceeds to the next step.
- If it is not, adds the monitor groups to the service's **/ordered_balgrp** object.

4. Adds the service balance to the balance monitors (**/balance_group/monitor** objects).

Removing Members from Hierarchy- and Paying Responsibility-Type Monitors

Use the PCM_OP_MONITOR_HIERARCHY_CLEANUP opcode to remove members from hierarchy-type or paying responsibility-type monitors when an account hierarchy changes. For example, this opcode is called when any of the following actions affect an account hierarchy:

- An account is removed from the hierarchy
- A child account changes from a *nonpaying* to a *paying* payment type
- The association between a balance group and a bill unit changes
- A line is removed from a child account

This opcode is triggered by the following events:

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

Important: Do not call this opcode directly. This opcode should be called through the event notification feature only. See "Configuring Event Notification for Balance Monitoring".

Displaying Balance Monitor Information in Client Applications

You can retrieve and display the following information in your client application:

- The balances for a monitor group
- A list of monitor groups owned by an account or service

Retrieving the Balances for a Monitor Group

To retrieve the total rolled-up balance for a monitor group, call the PCM_OP_BAL_GET_MONITOR_BAL opcode. The opcode retrieves either the balance monitor's current balance or the balance total for a specified time period.

The opcode returns one of the following, depending on whether you pass the PIN_FLD_DATE_BALANCES array in the input flist:

- If you pass the array with a start and end date, the opcode returns the total amount generated by members between those days.

  Note: The end date is exclusive. For example, if you enter January 1 as the start date and January 5 as the end date, the opcode sums all balance impacts generated on January 1, 2, 3, and 4.

- If you do not pass the array, the opcode returns the balance monitor's current rolled-up balance.

This opcode performs the following tasks to retrieve the balances:

1. Retrieves the POID of the **/balance_group/monitor** object and, optionally, the dates for which the balances are requested.
2. Reads the **/balance_group/monitor** object and retrieves the balance information.
3. When the input flist contains dates, retrieves the correct bucket for the specified dates from the sub-balances.
4. Returns the balance for the specified monitor group.

Retrieving the Balance Monitors Owned by an Account or Service

To retrieve a list of balance monitors owned by an account or service, call the PCM_OP_BAL_GET_ACCT_MONITORS opcode.

This opcode takes as input the POID of the **/account** or **/service** object and performs the following tasks:

1. Searches for all the **/group/sharing/monitor** objects associated with the account or service specified in the input flist.
2. For each **/group/sharing/monitor** object associated with the account or service, retrieves the **/balance_group/monitor** objects.
3. Returns a list of monitor groups owned by the specified account or service.

Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications

When a credit limit or threshold is reached, BRM generates a notification event containing information about the affected account and the reason for the breach. (See "About Using Event Notification to Alert Customers".)

When you run the **pin_monitor_balance** utility to update monitored balances with the current monitor impacts, it calls the PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode. This opcode updates the **/balance_group/monitor** objects and also compares the monitored balance with the credit limit and threshold values stored in the **/config/credit_profile** object. It generates notification events when the credit limits or thresholds are crossed.

For more information on the **pin_monitor_balance** utility, see "Running pin_monitor_balance to Update Monitored Balances" and "pin_monitor_balance".

The PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode generates two kinds of notification events:

- **/event/notification/threshold** when the balance crosses above the credit threshold
- **/event/notification/threshold_below** when the balance falls below the credit threshold

Important: These are notification events only and are not recorded in the database. See "About Notification Events" in *BRM Developer's Guide*.

Table 10–3 lists the fields in the notification events.

Table 10–3 Notification Event Fields for Monitoring

Field Name	**Description**
PIN_FLD_RESOURCE_ID	The resource ID.
PIN_FLD_AMOUNT	The balance impact for this event.
PIN_FLD_BAL_GRP_OBJ	The POID of the **/balance_group/monitor** object.
PIN_FLD_PERCENT	The percentage amount crossed.
PIN_FLD_SOURCE_OBJ	Source of the breach, for example, the POID of the event that caused the breach.
PIN_FLD_ALERT_TYPE	The alert type: credit limit or threshold percentage.
PIN_FLD_MONITOR_TYPE	Type of monitor: ▪ Paying responsibility credit exposure ▪ Hierarchy credit exposure ▪ Service level ▪ A type that you have defined.
PIN_FLD_REASON	The reason for the breach: ▪ Upward breach ▪ Downward breach ▪ Upward reset ▪ Downward reset ▪ Unknown reason
PIN_FLD_THRESHOLD	Tracks when the balance of a monitor group crosses a 5% boundary. For example, it tracks whether the current balance is 5%, 10%, or 15% of the credit limit.
PIN_FLD_CREDIT_THRESHOLDS	An array of credit thresholds and limits breached by this event.

Important: The opcode generates only one event if multiple thresholds are crossed. However, if multiple monitors cross the thresholds because of a single balance impact, the utility generates an event for each monitor.

You can use the information in the events to notify the CSR through a client application or the customer through email, for example, by writing a custom application.

For more information about using event notification, see the following topics:

- "Using Event Notification" in *BRM Developer's Guide*
- About Using Event Notification to Alert Customers
- Configuring Event Notification for Balance Monitoring

Example of Credit Threshold Notification Event Generation

Consider a balance monitor for an account or service with:

- A credit floor of $0
- A credit limit of $100
- Thresholds set at 25%, 75%, and 90%

- A current balance of $50

When a new event comes in with a balance of $26, the new balance is set to $76, and a notification event with the following information is generated:

```
PIN_FLD_ALERT_TYPE   Threshold
PIN_FLD_REASON   Upward breach
PIN_FLD_CREDIT_THRESHOLDS   75%
```

11 Accounts Receivable Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) accounts receivable utilities.

load_pin_ar_taxes

Use this utility to load configurable tax information into the **/config/ar_taxes** object in the BRM database.

This information enables BRM to correctly calculate tax reversals for particular event types during adjustments, disputes, and settlements. Typically, you define this type of tax information in the *BRM_Home***/sys/data/config/pin_config_ar_taxes.xml** file, where *BRM_Home* is the directory in which BRM is installed. You can optionally place the file in a different location or use a different file name.

For information on the contents and format of the **pin_config_ar_taxes.xml** file, see "Configuring Taxes for Adjustments, Disputes, and Settlements".

Note: You cannot load separate **/config/ar_taxes** objects for each brand. All brands use the same object.

Caution: The **load_pin_ar_taxes** utility overwrites the existing tax information that BRM uses for the various adjustments and dispute events types. If you are updating the current tax information base to add event types or change the tax specifications for an event type, you cannot load the new tax information only. You must load complete sets of tax information for all event types you want to treat as taxable each time you run the **load_pin_ar_taxes** utility.

Important: To connect to the BRM database, **load_pin_ar_taxes** requires a configuration file in the directory from which you run the utility. The **pin.conf** file for this utility is in *BRM_Home***/apps/pin_ar_taxes**. Also, you must restart the Connection Manager (CM) to make new tax calculation information available. For more information on the configuration file and restarting CM, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_ar_taxes [-v] [-h] -f input_file.xml
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

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

load_pin_ar_taxes *other_parameter* **–v >** *filename*.**log**

-h
Displays the utility's syntax and parameters.

-f *input_file.xml*
Specifies the name and location of the file that defines the configurable tax information; for example, **C:\taxconf\adjust_taxes.xml**. If you do not specify the **-f** parameter and a path, **load_pin_ar_taxes** looks for a file named **pin_config_ar_taxes.xml** in the directory from which you started the utility. You can use the *BRM_Home*/**sys/data/config/pin_config_ar_taxes.xml** file as a basis for creating a configuration file that suits your needs.

pin_apply_bulk_adjustment

Use this BRM command-line utility to apply an adjustment across a broad range of accounts, such as all accounts charged a regular day-time rate when they should have been charged a holiday rate.

This utility reads information on how to apply the bulk adjustment from a CSV format text file that you supply. The CSV file specifies the account POIDs and balance groups that BRM will adjust, the amount of the adjustment, and whether the adjustment should include taxes.

Note: To connect to the BRM database, **pin_apply_bulk_adjustment** needs a configuration file in the *BRM_Home*/**apps/pin_bulk_adjust** directory. You can define the name and level of the log file generated by this utility in the configuration file. For information on creating the configuration file, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

For information on the contents and format of the CSV file, see "About Adjusting Multiple Accounts Simultaneously".

Location

BRM_Home/**bin**

Syntax

```
pin_apply_bulk_adjustment [-v] [-h] -f input_file.csv
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

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

pin_apply_bulk_adjustment *other_parameter* **–v >** *filename*.**log**

-h
Displays the utility's syntax and parameters.

-f *input_file*.csv
Specifies the name and location of the file that specifies how BRM will apply the bulk adjustment; for example, **C:\bulkadj\bulk_run_1.csv**.

Results

Each account adjustment initiated during a bulk adjustment is handled as a separate transaction. Therefore, if any single adjustment in the bulk adjustment fails, BRM does not need to roll back the entire bulk adjustment. Rather, the adjustment remains in place for all accounts that had successful adjustments, and the utility identifies the accounts that failed to adjust so you can correct the problem. The utility provides this feedback in a log file (**default.pinlog**) that it generates in its start directory or in a directory specified by the configuration file.

pin_mass_refund

This utility creates a refund item for accounts that have a credit balance.

Refund items are used by the **pin_refund** utility to give refunds, and by Payment Tool to close refunds made to accounts that use the invoice payment method.

- For information about refunds, see "About Refunds".
- For more information about the **pin_mass_refund** utility, see "About Refunds".

Note: To connect to the database, the **pin_mass_refund** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*s.

Important: For multischema systems, you must run the utility separately against each database schema in your system. See "Running Non-MTA Utilities in Multischema Systems" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
pin_mass_refund   -active | close | inactive
                  [-pay_type payment_method]
                   [-test] [-verbose] [-help]
```

Parameters

-active | close | inactive
Specifies the status of the accounts for which to create refund items.

-pay_type *payment_method*
Specifies the payment method. There are three possible values:

- **10003** for credit card
- **10005** for direct debit
- **10018** for SEPA

Payment methods are defined in *BRM_Home*/**include/pin_pymt.h**.

-test
Finds the bill units that meet the criteria but does not create any refund objects.

-verbose
Displays information about successful or failed processing as the utility runs.

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

pin_mass_refund *other_parameter* **–verbose >** *filename*.**log**

-help
Displays syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_monitor_balance

Use this utility to update monitored balances and to check whether the balances have crossed thresholds.

For more information on using the **pin_monitor_balance** utility, see "Running pin_ monitor_balance to Update Monitored Balances".

Note: To connect to the BRM database, this utility needs a configuration file in the directory from which you run it. The **pin.conf** file for this utility is in *BRM_Home***/apps/pin_monitor**. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_Home***/bin**

Syntax

```
pin_monitor_balance [-d] [-v] [-t] [-h]
```

Parameters

-d
Prints error logs for debugging.

-v
Displays detailed information on status and error messages as the utility updates balances and generates notification events.

-t
Runs a test to find out how many accounts meet the criteria without performing the action. The test has no effect on the accounts. This is most useful when run with the **-v** option.

-h
Displays the syntax and parameters for this utility.

Results

This utility generates notification events when a balance crosses a credit threshold. It logs errors in the log file, which is either in the directory from which the utility was started or in a directory specified in the configuration file.

pin_refund

This utility finds accounts that have refund items and makes online refund transactions.

When you use multiple payment processors, you run this utility for each one. See "Using More Than One Payment Processor" in *BRM Configuring and Collecting Payments*.

For information about refunds, see "About Refunds".

Note: To connect to the BRM database, the **pin_refund** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Important: For multischema systems, you must run the utility separately against each database schema in your system. See "Running Non-MTA Utilities in Multischema Systems" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
pin_refund   -pay_type payment_type_indicator
             [-vendor] payment_processor_name
             [-active | close | inactive]
             [-verbose] [-test] [-help]
```

Parameters

-pay_type *payment_type_indicator*
Specifies the payment type. There are three possible values:

- **10003** for credit card
- **10005** for direct debit
- **10018** for SEPA

Payment types are defined in *BRM_Home*/**include/pin_pymt.h**.

-vendor *payment_processor_name*
Specifies the credit card processor or automated clearing house (ACH) to use for validating credit cards, debit cards, and direct debit transactions. This parameter is used to get the payment processor information from the **/config/ach** object.

This parameter is not applicable for the SEPA payment type.

For information on configuring payment processor information, see "Setting Up Merchants and Payment Processors" in *BRM Configuring and Collecting Payments*.

-active | close | inactive
Specifies the status of the accounts to give refunds to.

-verbose
Displays information about successful or failed processing as the utility runs.

To redirect the output to a log file, use the following syntax with the **-verbose** parameter. Replace *filename***.log** with the name of the log file:

pin_refund –verbose > *filename***.log**

Note: If a file with the same name exists, it is overwritten.

-test
Returns the number of refund items that would be created, but does not create any refund items.

-help
Displays syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

When it is called internally by the **pin_bill_day** script, the **pin_refund** utility logs error information in the **pin_billd.pinlog** file.