Oracle® Communications Billing and Revenue Management

Rerating Events

Release 15.1
F93194-01
April 2025

Oracle Communications Billing and Revenue Management Rerating Events, Release 15.1

F93194-01

Contents

Preface

Audience vi
Documentation Accessibility vi
Diversity and Inclusion vi

1 About Rerating Events

About Rerating Events 1-1
Run-time Options for Rerating 1-2
How BRM Applies the Balance Impacts of Rerating 1-2
- About Rerating Unbilled Events 1-3
  - Adjustments When Billing Follows Rerating 1-3
- About Rerating Billed and Posted Events 1-4
- Determining the G/L Entry for an Event 1-4
- How Rerating Affects Account Migration 1-4

How BRM Generates Rerated Events 1-4
- How Failed Rerate Jobs Are Processed 1-6

2 Configuring BRM to Rerate Events

About Setting Up Rerating 2-1
Configuring BRM to Rerate Events through ECE 2-1
Configuring Rerating Adjustment Events 2-2
Configuring How to Allocate Rerating Results 2-3
Enabling Deferred Tax Calculation During Rerating 2-4

3 Configuring BRM to Create Rerate Jobs Automatically

About Creating Rerate Jobs Automatically 3-1
Creating Rerate Jobs for Backdated Events 3-2
Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Charges 3-3
Creating Rerate Jobs for Rollover Corrections 3-4
Enabling Rerating for Canceled Noncurrency Resources 3-7
Creating Rerate Jobs for Midsession Rating Changes 3-8

Configuring Event Notification for Automatic Rerate Job Creation 3-9
Assigning Reason Codes During Automatic Job Creation 3-10
Allowing Rerating of Old Events 3-10

4 Manually Creating Rerate Jobs

About Manually Selecting Events for Rerate Jobs 4-1
Selecting Events for Rerate Jobs 4-2
- Selecting Events for a Single Account 4-2
- Selecting Events for a List of Accounts 4-2
- Selecting Events for a Sharing Group 4-3
- Selecting Events Associated with Offers 4-3
- Selecting Events Based On Event Type 4-4
- Selecting Events Based On Service Type 4-5
- Selecting Events Based On Subscriber Service ID 4-5
- Selecting Events Associated with Balance Groups 4-5
- Selecting Events Based on a Custom Event Field 4-6

Backing Out Rating for Select Events 4-7
Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Events 4-8
Assigning Reason Codes to Rerate Jobs 4-8
Getting an Estimated Number of Rerated Events 4-9
Specifying the Event Sequence in a Rerate Job 4-9

5 Processing Events in Rerate Jobs

About the Rerating Process 5-1
About Processing Rerate Jobs 5-2
- Steps for Processing Rerate Jobs 5-2
- Generating Reports During Rerating 5-2
- Recycling Records Suspended During Rerating 5-3
- Processing Failed Account Events in Rerate Jobs 5-4
- Purging Rerate Jobs 5-4

Specifying the Order to Process Rerate Jobs 5-5

6 Tuning Rerating Performance

Improving pin_rerate Performance 6-1
Setting the Rerating Event Cache Size (Fetch Size) 6-1
Configuring the Number of Accounts Per Job and Number of Jobs per Transaction 6-2

7 Rerating Utilities

load_pin_rerate_flds 7-1
pin_rate_change 7-2
pin_rerate 7-3
- Rerate Job Creation Options 7-4
- Rerate Job Processing Parameters 7-7
- Rerate Job Configuration Options 7-8

Preface

This guide describes how to configure and run the event rerating process in Oracle Communications Billing and Revenue Management.

Audience

This guide is intended for system administrators.

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

1

About Rerating Events

Learn about rerating events in Oracle Communications Billing and Revenue Management (BRM) using Oracle Communications Elastic Charging Engine (ECE).

Topics in this document:

- About Rerating Events
- Run-time Options for Rerating
- How BRM Applies the Balance Impacts of Rerating
- How BRM Generates Rerated Events

If you are using the BRM real-time rating and batch rating engines for usage rating, see *Oracle Communications Billing and Revenue Management Rerating Pipeline Events* for information about rerating events.

About Rerating Events

You might need to rerate events for several reasons. For example:

- To correct charges.

  For example, if several customers are charged the wrong price for using a service, you can rerate the events for all those customers.

- To rerate an account when a CSR backdates a subscriber's purchase or cancellation.

  For example, if a customer upgrades a service and purchases a backdated charge offer, you can rerate events based on the old charge offer using the upgraded charge offer.

- To rerate events that should have been rated using a different charge offer.

- To correct rollover amounts consumed by offline charging.

  This can occur if a delayed event arrives after the end of the accounting cycle and during the delayed billing period. The event can borrow against the rollover of the current cycle even when events of the current cycle consume the current rollover.

- To reset first-usage validity dates for recurring charges associated with offers or balances that start on the first usage.

  For example, suppose the first event rated was not the first event to use the offer's service. In that case, rerating corrects the order of events and resets the validity period based on the first-usage event.

- To perform backout-only rerating.

  Backout-only rerating backs out the balance impacts of rating without reapplying new balance impacts.

BRM rerates events in two separate processes:

1. It finds the events to rerate and groups them into a rerate job. For example, it could find all events associated with a charge offer and then group them into a rerate job.

   You can perform this task in two different ways:

- Manually using the **pin_rerate** utility. See "Manually Creating Rerate Jobs".
- Automatically through the BRM automatic rerating mechanism. In this case, when an event occurs, such as backdating a customer's purchase, BRM creates rerate jobs to rerate those events. You must configure BRM to create rerate jobs automatically. See "Configuring BRM to Create Rerate Jobs Automatically".

2. It rerates the events in the rerate job. You perform this task by running the **pin_rerate** utility manually or through a cron job. See "Processing Events in Rerate Jobs".

   If you use ECE for usage rating, the BRM server sends the usage events to ECE to rerate. ECE rerates the events and sends the new balance impacts to the BRM server.

   If you use the BRM real-time rating and batch rating engines for usage rating, BRM rerates the events. See *Oracle Communications Billing and Revenue Management Rerating Pipeline Events*.

BRM uses rerate jobs for two purposes:

- To manage the processing of many events in an efficient, transactional way. Multiple rerate jobs are created for every rerate request.
- To manage the status of the accounts selected for rerating. For example, if an account is part of a rerate job in progress, it cannot be migrated to another database until rerating has finished.

Run-time Options for Rerating

When you rerate events, you have the following options:

- You can select events for rerating based on various criteria such as the time the event occurred, the account number, bundles, charge offers, discount offers, event, service, and so on. See "About Manually Selecting Events for Rerate Jobs" for information.
- You can control the order in which rerating occurs by creating reason codes. For example, to rerate events that impact future rating, such as volume-based discounting, use reason codes to rerate those events first. See "Specifying the Order to Process Rerate Jobs".
- You can specify the order in which events are rated by either the order that they were created in the BRM database, or by the order in which the events occurred. See "Specifying the Event Sequence in a Rerate Job" for information.
- You can run reports during rerating that include statistics such as the original amount, rerated amount, and the difference between them. See "Generating Reports During Rerating" for information.
- You can rerate events without reapplying a charge. This is known as back-out rerating. You typically use this only for usage charges that should not have occurred. See "Backing Out Rating for Select Events" for information.

How BRM Applies the Balance Impacts of Rerating

BRM rerates both billed and unbilled events and can rerate events that belong to multiple billing cycles. Therefore, BRM might need to distribute balance impacts across numerous balances.

When rerated events include noncurrency balance impacts, BRM rerating charges or credits the account accordingly. Financial effects on accounts receivable (A/R), general ledger (G/L), and taxation are considered during rerating.

When an event is rerated, BRM backs out the event from the BRM database by creating an adjustment event that entirely negates the original balance impacts. Adjustments are handled differently depending on if the event is billed or unbilled.

- If the event is unbilled, a shadow adjustment event is created. This event is called a *shadow event* because its balance impact is added to the original event's bill item rather than an adjustment item. See "About Rerating Unbilled Events".
- If the event has already been billed or posted to the G/L, a regular adjustment event is created. See "About Rerating Billed and Posted Events".

When the total rerating adjustment is zero, BRM does not generate a rerating adjustment event if the balance impacts of rerating and previous rating are equivalent. To determine if the balance impacts are equal, BRM compares the values of certain balance impact fields for rerating with the previous rating, such as the balance element ID and balance group. You can customize how BRM determines whether the balance impacts of rerating and the previous rating are equivalent by modifying the event balance impact fields used for comparison. See "Configuring Rerating Adjustment Events".

About Rerating Unbilled Events

When unbilled events are rerated, a shadow adjustment event is added to the original event's bill item instead of an adjustment item.

> **Note:**
>
> When recording a shadow event, you can customize your invoice to either show the details of rerating or show the result only in an end balance. See *Oracle Communications Billing and Revenue Management Designing and Generating Invoices* for information.

Shadow events are handled differently for usage events and recurring events:

- When unbilled usage events are rerated, the shadow adjustment event fully negates the original balance impacts and applies new balance impacts for the rerated amount.
- When unbilled recurring events are rerated, the shadow adjustment event fully negates the original balance impacts. Then a new recurring event is created that applies the rerated balance impacts. The new recurring event is the same type of event as the original event.

Adjustments When Billing Follows Rerating

When the purchase start time is the same as the current accounting cycle end date, the cycle forward fee balance impact is applied to the next month's bill instead of the current month's bill when billing is run after rerating is performed.

For example, suppose an account is created on January 1 with a cycle forward fee of $20, and the purchase start time is deferred by one month (set to the accounting cycle end date). When **pin_rerate** is run on February 1, rerating internally triggers automatic billing because the purchase start time is the same as the accounting cycle end date. The billing process changes the status of the bill item to open. Because the bill item status is now open, rerating applies the rerate adjustments to the next bill. When regular billing is run on February 2, the cycle fee is not applied to the January bill and is instead applied to the February bill.

About Rerating Billed and Posted Events

When you rerate events that have already been billed, and unbilled events already posted to the G/L, an adjustment event entirely negates the original balance impacts. It applies new balance impacts for the rerated amount. The results of rerating are applied to the adjustment item in the next bill, which is the bill for the current cycle. For billed cycle fee, fold, rollover, and cycle discount events, a new cycle event is then created that applies the rerated balance impacts.

When billing is run, if a noncurrency balance is zero, no cycle event is generated for that balance. For example, no rollover event is created if there are no remaining minutes to roll over at billing time. However, if rerating creates a nonzero balance that was previously zero when billed, the rerating process generates the appropriate cycle event.

Determining the G/L Entry for an Event

The general ledger (G/L) entry is determined at rating time. If you rerate due to pricing changes, the G/L entry could change. Whether to record a shadow or adjustment event, you must determine the correct balance to be posted in the G/L.

When recording a shadow event, the G/L ID of the rerating balance impacts has the same G/L ID as the original event's balance impacts. If you changed the G/L ID after the original event was rated, the balance impacts of rerating use the new G/L ID.

When recording an adjustment, you can configure the G/L ID to use. For example, you can use the same G/L ID as the original event, a new G/L ID, or an adjustment G/L ID (a separate G/L ID bucket to record all adjustments as part of rerating). The adjustment G/L ID can be the same or different than the G/L ID used for regular (not rerated) event adjustments. For information about setting up G/L IDs, *BRM Collecting General Ledger Data*.

How Rerating Affects Account Migration

When an account is selected for rerating, the account cannot be migrated to another database until rerating is complete. Otherwise, the account is not rerated. For this reason, the Account Migration Manager (AMM) does not allow the migration of an account to another database if it is being rerated.

However, the account is migrated if the rerate job status is NEW. In this case, the AMM deletes the account from the rerate job in the source database and creates a new rerate job with the account in the destination database.

> **Note:**
>
> After the account migration, you must run **pin_rerate** in the destination database to rerate the account.

How BRM Generates Rerated Events

BRM rerates events in chronological order:

- Usage events are rerated in order starting with the earliest usage event.

- Rollover, fold, cycle discount, and cycle fee events are generated in the order they logically occur. For example, billing-time discount events are generated before rollover events.

There are three ways in which rerating generates events:

- Rerate the original event.

  This process passes the event being rerated to ECE for rerating. This is the most basic rerating process.

- Reapply business logic based on information in the original event.

  This process passes information from the event being rerated to the opcode that generated the event. The opcode may or may not generate a new event. This process is used when the event attributes may be different after rerating. For example, a charge offer's purchase fee may have been changed since the purchase fee event was generated.

  Usually, purchase events and cancellation events are not directly rerated. Instead, information in the original purchase event and cancel event is used to generate new fee events, if applicable.

- Reapply business logic independent of the original event.

  This process performs business logic that might generate new events even if no existing event exists. With this process, information from the original event is not used. Instead, the original event's business logic is reapplied, and a new event is generated, if appropriate.

  For example, when rerating an account for a specific period, if the events selected for rerating are associated with charge offers that have a cycle fee, rerating calls the cycle fee opcode and generates a new cycle fee event based on the charge offer's current rates. This is done independent of any existing cycle fee event. If there is an existing cycle fee event, it is replaced by the corresponding new cycle fee event generated by rerating.

  In some cases, the purchase- and cancellation-fee events are directly rerated because the purchase and cancellation events may not be available for rerating. For example, when rerating deferred purchase fees or when using selective rerating in which only purchase or cancellation-fee events are specified.

The following events are not rerated but are reapplied because they were not originally generated by rating:

- **/event/billing/adjustment**
- **/event/billing/debit**
- **/event/billing/dispute**
- **/event/billing/item**
- **/event/billing/settlement**
- **/event/billing/payment**
- **/event/billing/reversal**
- **/event/billing/writeoff**
- **/event/billing/cycle/tax**
- **/event/billing/refund**
- **/event/billing/charge**

Table 1-1 shows the rerating process for specific types of events.

Table 1-1 Rerating Process and Event

Event Type	Order of Rerating	How Rerated
Usage	Chronological, based on event time.	Rerate the original event.
Cycle fee	Generated according to the time that the charge offer is purchased or canceled or when the associated accounting cycle starts or ends, and based on the logical order of events.	Reapply business logic independent of the original event.
Rollover	Generated according to the time that the associated billing cycle ends, and based on the logical order of events.	Reapply business logic independent of the original event.
Fold	Generated according to the time that the associated billing cycle ends, and based on the logical order of events.	Reapply business logic independent of the original event.
Billing-time discount	Generated according to the time that the associated billing cycle ends, and based on the logical order of events.	Reapply business logic independent of the original event.
Purchase	Chronological, based on event time.	Reapply business logic based on information in the original event.
Cancellation	Chronological, based on event time.	Reapply business logic based on information in the original event.

How Failed Rerate Jobs Are Processed

The accounts in a rerate job are typically processed individually in separate rerate operations. When rerating fails for one or more accounts in a rerate job, the **pin_rerate** utility sets the status of the accounts in the rerate job batch to FAILED. When the rerating process is complete for the rerate job, **pin_rerate** creates a new rerate job consisting of only the accounts that failed. The new rerate job is processed the next time **pin_rerate** is run.

If an account selected for rerating is associated with a subscription service that was transferred during the period for which rerating is specified, then the account to which the service was transferred is included in the rerate job and all those accounts are rerated concurrently in a single rerate operation. When rerating fails for one of these accounts, then rerating fails for all accounts in the rerate request. In this case, **pin_rerate** creates a new rerate job containing all the accounts in the rerate request.

For example, subscription service X is originally owned by Account A and transferred to Account B on June 15. Later in the month, it is transferred from Account B to Account C. Rerating of Account A from June 1 also results in rerating of accounts B and C. Accounts A, B, and C are grouped together in a single rerate request. If rerating fails for any of these accounts, **pin_rerate** creates a new rerate job consisting of all three accounts in a single rerate request. The accounts are rerated again the next time **pin_rerate** is run.

2

Configuring BRM to Rerate Events

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) to enable rerating.

Topics in this document:

- About Setting Up Rerating
- Configuring BRM to Rerate Events through ECE
- Configuring Rerating Adjustment Events
- Configuring How to Allocate Rerating Results
- Enabling Deferred Tax Calculation During Rerating

About Setting Up Rerating

To set up BRM and ECE to rerate events:

1. Configure BRM to rerate events through ECE. See "Configuring BRM to Rerate Events through ECE".
2. (Optional) Customize how BRM determines whether rerating created different billing results. See "Configuring Rerating Adjustment Events".
3. (Optional) Configure BRM to allocate adjustment items to their original bill during rerating. See "Configuring How to Allocate Rerating Results".
4. (Optional) Configure BRM to apply deferred taxes during the rerating process. See "Enabling Deferred Tax Calculation During Rerating".

Configuring BRM to Rerate Events through ECE

To configure BRM to rerate events through ECE:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) in a text editor.
2. Ensure this entry is in the file:

   ```
   - cm fm_module BRM_home/lib/fm_rerate.so fm_rerate_config - pin
   ```

3. Set the **em_group** entry to **PCM_OP_RATE_ECE_EVENT**:

   ```
   - cm em_group rerating PCM_OP_RATE_ECE_EVENT
   ```

4. Edit the **em_pointer** entry to match your ECE EM Gateway configuration:

   ```
   - cm em_pointer rerating ip em_gateway_host em_gateway_port
   ```

   where:

   - *em_gateway_host* is the IP address of the machine on which the ECE EM Gateway resides.
   - *em_gateway_port* is the port of the machine on which the ECE EM Gateway resides.

5. Save and close the file.
6. Stop and restart the CM.

Configuring Rerating Adjustment Events

When you rerate usage events, BRM creates an adjustment event to apply the balance impacts of rerating. The adjustment's balance impact is the difference between the previous rating results and the rerated results.

When the results are the same, BRM does not create an adjustment event. However, even when the results are the same, there might be differences in the rating and rerating results in some of the individual balance impacts contributing to the total charge. Therefore, before it can determine that rerating resulted in no overall change in the balance impact, BRM must ensure that the balance impacts in each rating process are equivalent.

To determine whether the balance impacts of rerating and the previous rating are equivalent, BRM checks a list of values in all balance impacts. For example, BRM determines whether the same charge offer was used for all balance impacts. If the balance impact fields have the same values in the event for both rerating and the previous rating, and the total rerating adjustment is zero, the results of rerating and the previous rating are considered equivalent and no adjustment event is created.

BRM uses the values in the following fields to check the rerating and previous rating results, but you can configure BRM to use additional fields:

- The ID of the impacted balance element
- The ID of the impacted balance group
- The G/L ID
- The tax code
- The charge offer used to rate the event

To configure BRM to use additional fields, modify the **pin_rerate_compare_bi.xml** file and then run the **load_pin_rerate_flds** utility to load the contents of the file into the **/config/rerate_flds/compare_bi** object in the BRM database.

> **Note:**
>
> The **load_pin_rerate_flds** utility overwrites existing balance-impact comparison fields. If you are updating balance-impact comparison fields, you cannot load new fields only. You must load complete sets of the balance-impact comparison fields when you run the utility. See "load_pin_rerate_flds" for more information.

To customize the fields used to compare the event balance impacts of rerating with the previous rating:

1. Open the *BRM_home***/sys/data/config/pin_rerate_compare_bi.xml** file in a text editor.
2. Add a **RerateCompareBalImpacts** element for each event balance impact field. The following balance impact fields are mandatory:
   - PIN_FLD_RESOURCE_ID
   - PIN_FLD_BAL_GRP_OBJ

- PIN_FLD_GL_ID
- PIN_FLD_TAX_CODE

> **Note:**
>
> PIN_FLD_PRODUCT_OBJ is specified in the **pin_rerate_compare_bi.xml** file by default but is an optional field.

You can specify any additional field from the **/event** object's PIN_FLD_BAL_IMPACTS array.

3. Save and close the file.
4. Load the contents of the XML file into the **/config/rerate_flds/compare_bi** object:

   ```
   load_pin_rerate_flds BRM_home/sys/data/config/pin_rerate_compare_bi.xml
   ```

   For more information, see "load_pin_rerate_flds".
5. Stop and restart the CM.
6. Verify that the balance-impact comparison fields were loaded by displaying the **/config/rerate_flds/compare_bi** object by using Object Browser, or by using the **robj** command with the **testnap** utility.

Configuring How to Allocate Rerating Results

By default, BRM creates unallocated items for any adjustment items created due to rerating. You can configure BRM to instead allocate adjustment items to their original bill.

- For open item accounting, the adjustment items are allocated to each bill that included events rerated and resulted in adjustments.
- For balance forward accounting, adjustment items are allocated to the last bill only.

If BRM has rerated a **/bill** object without allocating automatic adjustments (from the rerating) to the original bills, the corrective bill for that **/bill** object does not include the item adjustments generated by that rerating. If you enable the **AllocateReratingAdjustments** business parameter and rerun the rerating process for the bill, BRM does not allocate the adjustments. You must manually allocate the rerated items *before* you generate the corrective bill for such a **/bill** object.

To allocate adjustment items to their original bill during rerating:

1. Go to the *BRM_home***/sys/data/config** directory.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates an XML file named **bus_params_rerate.xml.out** in your working directory.
3. Open the **bus_params_rerate.xml.out** file.
4. Set the **AllocateReratingAdjustments** parameter to **enabled**:

   ```
   <AllocateReratingAdjustments>enabled</AllocateReratingAdjustments>
   ```
5. Save and rename the file as **bus_params_rerate.xml**.

6. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_rerate.xml
   ```

7. Stop and restart the CM.

Enabling Deferred Tax Calculation During Rerating

By default, BRM calculates taxes on any deferred taxable amount in rerated events during the subsequent bill run, resulting in rerated taxes appearing on the next invoice.

You can configure BRM to apply deferred taxes during the rerating process instead, allowing deferred tax amounts to appear on corrected invoices.

To enable the calculation of deferred taxes during rerating:

1. Go to the *BRM_home*/**sys/data/config** directory.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates an XML file named **bus_params_rerate.xml.out** in your working directory.
3. Open the **bus_params_rerate.xml.out** file.
4. Set the **ApplyDeferredTaxDuringRerating** parameter to **enabled**:

   ```
   <ApplyDeferredTaxDuringRerating>enabled</ApplyDeferredTaxDuringRerating>
   ```

5. Save and rename the file as **bus_params_rerate.xml**.
6. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_rerate.xml
   ```

7. Stop and restart the CM.

3

Configuring BRM to Create Rerate Jobs Automatically

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) to automatically create rerate jobs when specified events occur in the system, such as events being backdated.

Topics in this document:

- About Creating Rerate Jobs Automatically
- Creating Rerate Jobs for Backdated Events
- Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Charges
- Creating Rerate Jobs for Rollover Corrections
- Enabling Rerating for Canceled Noncurrency Resources
- Creating Rerate Jobs for Midsession Rating Changes
- Configuring Event Notification for Automatic Rerate Job Creation
- Assigning Reason Codes During Automatic Job Creation
- Allowing Rerating of Old Events

About Creating Rerate Jobs Automatically

You can configure BRM to automatically create rerate jobs when specified events occur in your system. These rerate jobs are processed the next time you run the **pin_rerate** utility. If you create rerating jobs automatically, you should run the **pin_rerate** utility from a cron job.

To configure BRM to create rerate jobs automatically:

1. Configure BRM to automatically create rerate jobs for backdated events. See "Creating Rerate Jobs for Backdated Events".
2. Configure BRM to automatically create rerate jobs when cycle forward and cycle forward arrears charges change. See "Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Charges".
3. Configure BRM to automatically create rerate jobs for rollover corrections. See "Creating Rerate Jobs for Rollover Corrections".
4. Configure BRM to automatically create rerate jobs when products with noncurrency resources are canceled. See "Enabling Rerating for Canceled Noncurrency Resources".
5. Configure BRM to automatically create rerate jobs for midsession-rating changes. See "Creating Rerate Jobs for Midsession Rating Changes".
6. Configure BRM to trigger notification events for automatic rerating. See "Configuring Event Notification for Automatic Rerate Job Creation".
7. (Optional) Customize BRM to assign reason codes to each automatically created rerate job. See "Assigning Reason Codes During Automatic Job Creation".

8. (Optional) Configure BRM to rerate backdated events older than the allowable date. See "Allowing Rerating of Old Events".
9. (Optional) Customize BRM to automatically create rerate jobs for custom events, such as when a charge offer is canceled. To do so, customize the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode. See "Customizing Automatic Creation of Rerate Jobs" in *BRM Opcode Guide*.
10. (Optional) Allow BRM to rerate events that are older than the defined number of billing cycles. See "Allowing Rerating of Old Events"

Creating Rerate Jobs for Backdated Events

You can configure BRM to automatically create rerate jobs when you backdate the following events:

- The purchase or cancellation of a charge offer, discount offer, or bundle.
- An extended rating attribute (ERA) modification.

  For example, subscribers change their service-level agreement from Silver to Gold on July 12. The CSR backdates the change to July 1 to let subscribers apply the Gold-level benefits to all usage for July. BRM rerates all relevant events for those accounts that occurred between July 1 and the current time.
- An adjustment to a noncurrency balance.

  For example, a subscriber's billing day is the first day of the month. On February 15, the balance of free minutes is adjusted from 100 to 500, and the adjustment is backdated to January 15. BRM rerates all relevant events from the account that occurred between January and the current time. Billing events that previously occurred on February 1, such as billing-time discounts, rollovers, and folds, are recalculated based on the backdated amount of minutes.

To configure BRM to automatically create rerate jobs for backdated events:

1. Go to *BRM_home*/**sys/data/config**.
2. 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 an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**, set **BackdateTriggerAutoRerate** to enabled:

   ```
   <BackdateTriggerAutoRerate>enabled</BackdateTriggerAutoRerate>
   ```

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

4. In **bus_params_subscription.xml.out**, set **BackdateWindow** to the minimum number of seconds an event must be backdated to trigger rerating:

```
<BackdateWindow>value</BackdateWindow>
```

The default value is **3600**.

5. In **bus_params_subscription.xml.out**, set **NumBillingCycles** to the maximum number of billing cycles allowed between the current time and the backdated event date for rerating events:

   ```
   <NumBillingCycles>value</NumBillingCycles>
   ```

   The default value is **1**.

6. Save and exit the file.
7. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
8. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

The new values are available immediately.

Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Charges

> **Important:**
>
> Rerating recurring events can affect performance.

To enable BRM to automatically create rerate jobs to rerate cycle forward and cycle forward arrears events after they are charged:

1. Go to *BRM_home***/sys/data/config**.
2. 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 an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**, set **RateChange** to **enabled**:

   ```
   <RateChange>enabled</RateChange>
   ```

The default value is **disabled**.

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

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
7. Save and exit the file.
8. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
9. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

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

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

> **Note:**
>
> You must also configure BRM to generate notification events when rerate cycle forward and cycle forward arrears charges occur. See "Configuring Event Notification for Automatic Rerate Job Creation".

Creating Rerate Jobs for Rollover Corrections

If a delayed event arrives after the end of the accounting cycle and during the delayed billing period, rerating can allow it to consume the rollover of its proper cycle even when the rollover

had been consumed by events for the current cycle. You can configure BRM to automatically create rerate jobs when rollover corrections occur.

If rerating and rollover correction are both enabled and delayed events arrive, the current cycle events are rerated and the rollover is reallocated so that it comes from the appropriate cycles.

There are two business parameters that can be used to configure this processing. **RerateDuringBilling** is used to trigger rerating at the next billing date to ensure that customers are not charged more as a result of delayed CDRs. **RolloverCorrectionDuringBilling** can be used in conjunction with **RerateDuringBilling** to ensure that delayed CDRs consume the correct resources. (**RolloverCorrectionDuringBilling** has no effect unless **RerateDuringBilling** is enabled.)

The following examples display how these parameters affect resource consumption and billing.

Situation (all cases):

- The subscriber has 1000 free minutes per month, with rollover.
- A delay period of 5 days is enabled.
- No rollover minutes or delayed CDRs exist from May.
- CDRs for 700 minutes from June are received by the end of June.
- June 30/July 1 (midnight): 300 minutes roll over from June and 1000 new minutes are granted for July.
- July 2: CDRs for 500 minutes from July are received, which consume the 300 June rollover minutes and 200 of the July grant minutes.
- July 3: CDRs for 400 minutes from June are received, for a total of 1100 minutes used in June.
- No further minutes are used in July.

Case 1: Both **RerateDuringBilling** and **RolloverCorrectionDuringBilling** are **disabled**:

On July 2, the 500 minutes used in July consume the 300 June rollover minutes and 200 minutes from the July grant. On July 3, the CDRs for the additional 400 minutes for June are received, but the rollover minutes for June have already been consumed, and minutes used in June cannot consume minutes from the July grant, so the customer is billed for all 400 minutes.

- Customer billed minutes for June: 400
- June grant minutes consumed: 700
- June rollover minutes consumed: 300
- July grant minutes consumed: 200
- July minutes rolled over on July 31/August 1: 800

Case 2: **RerateDuringBilling** is **enabled** and **RolloverCorrectionDuringBilling** is **disabled**:

On July 2, the 500 minutes used in July consume the 300 June rollover minutes and 200 minutes from the July grant. On July 3, the CDRs for the additional 400 minutes for June are received. When the next bill is run, rerating is run. The consumption of the 300 June rollover minutes by the July CDRs is rolled back, and the CDRs for the delayed June minutes consume the June rollover minutes. The customer is billed for the remaining 100 minutes used in June. The 500 minutes from July consume a total of 500 minutes of the July grant.

- Customer billed minutes for June: 100
- June grant minutes consumed: 700

- June rollover minutes consumed: 300
- July grant minutes consumed: 500
- July minutes rolled over on July 31/August 1: 500

Case 3: Both **RerateDuringBilling** and **RolloverCorrectionDuringBilling** are **enabled**:

On July 2, the 500 minutes used in July consume the 300 June rollover minutes and 200 minutes from the July grant. On July 3, the CDRs for the additional 400 minutes for June are received. When the next bill is run, rerating is run. The consumption of the 300 June rollover minutes by the July CDRs is rolled back. The rollover of the 300 minutes from June is also rolled back and they are returned to the original June grant. The CDRs for 300 of the delayed June minutes consume those minutes as part of the original June grant. The customer is billed for the remaining 100 minutes used in June. The 500 minutes from July consume a total of 500 minutes of the July grant.

- Customer billed minutes for June: 100
- June grant minutes consumed: 1000
- June rollover minutes consumed: 0 (rollover is reversed)
- July grant minutes consumed: 500
- July minutes rolled over on July 31/August 1: 500

To create rerate jobs for rollover corrections:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the billing instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates an XML file named **bus_params_billing.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_billing.xml.out**, set **RerateDuringBilling** to enabled:

   ```
   <RerateDuringBilling>enabled</RerateDuringBilling>
   ```

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

4. (Optional) In **bus_params_billing.xml.out**, set **RolloverCorrectionDuringBilling** to enabled:

   ```
   <RolloverCorrectionDuringBilling>enabled</RolloverCorrectionDuringBilling>
   ```
5. Save and exit the file.
6. Rename the **bus_params_billing.xml.out** file to **bus_params_billing.xml**.
7. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

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

8. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

9. Stop and restart the CM.

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

> **Note:**
>
> You must also configure BRM to generate notification events when rollover corrections occur. See "Configuring Event Notification for Automatic Rerate Job Creation".

Enabling Rerating for Canceled Noncurrency Resources

To enable the automatic creation of rerate jobs when a product with noncurrency resources is canceled:

1. Go to *BRM_home*/**sys/data/config**.
2. 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 an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.

3. In **bus_params_subscription.xml.out**, set the **CreateRerateJobDuringCancel** parameter to **enabled**:

   ```
   <CreateRerateJobDuringCancel>enabled</CreateRerateJobDuringCancel>
   ```

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

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

8. Stop and restart the CM.

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

Creating Rerate Jobs for Midsession Rating Changes

You can configure rerating to account for midsession changes in the rating conditions. For example, a subscriber might initiate a Friends and Family discount during a call, which changes the pricing midsession. When the call is rerated:

- If you enable this option, rerating uses the original pricing from the call's start time until the called number is added to the Friends and Family list, and then uses the discounted pricing for the remaining session.
- If you do not enable this option, rerating uses the discounted pricing for the entire call.

To create rerate jobs for midsession rating changes:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the rerating instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

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

3. In **bus_params_rerate.xml.out**, set the **OfferEligibilitySelectionMode** parameter to **timeperiod**:

   ```
   <OfferEligibilitySelectionMode>timeperiod</OfferEligibilitySelectionMode>
   ```

   The default value is **endtime**.

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

4. Save and exit the file.
5. Rename the **bus_params_rerate.xml.out** file to **bus_params_rerate.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

```
pin_bus_params bus_params_rerate.xml
```

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

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

8. Stop and restart the CM.

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

> **Note:**
>
> You must also configure BRM to generate notification events when midsession-rating changes occur. See "Configuring Event Notification for Automatic Rerate Job Creation".

Configuring Event Notification for Automatic Rerate Job Creation

To configure notification events for automatic rerating:

1. Open the *BRM_home*/**sys/data/config/pin_notify** file in a text editor.
2. Ensure that the default automatic rerate job creation entries are included:

```
3787 0 /event/notification/auto_rerate
3787 0 /event/notification/rate_change
3787 0 /event/notification/rollover_correction/rerate
```

3. To trigger notification events for first usage events, add the following lines:

```
9071 0 /event/billing/product/fee/cycle/cycle_arrear
9071 0 /event/billing/product/fee/cycle/cycle_forward_annual
9071 0 /event/billing/product/fee/cycle/cycle_forward_arrear
9071 0 /event/billing/product/fee/cycle/cycle_forward_bimonthly
9071 0 /event/billing/product/fee/cycle/cycle_forward_monthly
9071 0 /event/billing/product/fee/cycle/cycle_forward_quarterly
9071 0 /event/billing/product/fee/cycle/cycle_forward_semiannual
3787 0 /event/notification/rollover_correction/rerate
```

4. To trigger notification events when a charge changes midcycle, add the following lines:

```
3768 0 /event/audit/price/product_update
3768 0 /event/audit/price/product_complete
```

5. To trigger notification events when a rollover event is corrected, add the following lines:

```
3787 0 /event/notification/rollover_correction/rerate
```

6. Save and close the file.
7. Run the **load_pin_notify** utility.

   See "Loading the Event Notification List" in *BRM Developer's Guide*.

Assigning Reason Codes During Automatic Job Creation

You can configure BRM to assign reason codes to jobs during the automatic rerate job creation process. To do so, you customize the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST opcode to automatically assign a custom reason code to each job. By default, the opcode only returns the input as output. For more information, see "Specifying a Rerate Reason Code" in *BRM Opcode Guide*.

Assigning reason codes to rerate jobs allows you to control the order in which jobs are processed. See "Specifying the Order to Process Rerate Jobs" for more information.

Allowing Rerating of Old Events

When you configure automatic rerate job creation, you use the **NumBillingCycles** business parameter to limit the backdated events that can be rerated to a specified number of billing cycles in the past. This setting also applies to manually rerating events. To manually rerate backdated events older than the allowable date, you enable the **AllowBackdateNoRerate** business parameter.

To enable rerating older events when they are backdated:

- Automatic rerate job creation must be enabled. See "Creating Rerate Jobs for Backdated Events" for instructions.
- Enable the **AllowBackdateNoRerate** business parameter according to the instructions below.
- After this option is enabled, run the **pin_rerate** utility manually, including the selection criteria necessary to rerate the events.

To enable rerating of old events:

1. Go to *BRM_home***/sys/data/config**.
2. 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 an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**, set the **AllowBackdateNoRerate** parameter to **enabled**:

   ```
   <AllowBackdateNoRerate>enabled</AllowBackdateNoRerate>
   ```

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

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.

6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

8. Stop and restart the CM.

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

4

Manually Creating Rerate Jobs

Learn how to create rerate jobs manually by running the **pin_rerate** utility in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- About Manually Selecting Events for Rerate Jobs
- Selecting Events for Rerate Jobs
- Backing Out Rating for Select Events
- Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Events
- Assigning Reason Codes to Rerate Jobs
- Getting an Estimated Number of Rerated Events
- Specifying the Event Sequence in a Rerate Job

About Manually Selecting Events for Rerate Jobs

You manually specify the events to include in a rerate job using the **pin_rerate** utility. The utility selects events for the rerate job based on criteria you specify, which consists of a timestamp and one of the following:

- One or more event types, such as purchase events and cancellations events
- One or more account numbers
- One or more charge offers, discount offers, or bundles
- One or more service types
- A sharing group owner
- One or more balance groups
- A customer's subscriber ID
- Custom criteria

BRM finds events created after the specified timestamp generated by the accounts that meet your criteria. It then groups the events into a rerate job.

For example, assume you specify to rerate events generated after 15 June for the GSM data service type (**/service/telco/gsm/data**). In this case, BRM does the following:

1. Finds all accounts associated with the GSM data service type.

   For example, it finds that accounts A, B, and C meet this criteria.

2. For the accounts that meet the criteria, finds all events that they generated after 15 June. This includes any event type generated by the account, not just purchase events.

   For example, it finds all purchase events, cycle forward events, product cancellation events, and so on that accounts A, B, and C generated after 15 June.

3. Adds the events from step 2 to a rerate job.

Selecting Events for Rerate Jobs

You select the events to rerate by running the **pin_rerate** utility with the job creation options. You can select events for rerate jobs based on the following:

- Selecting Events for a Single Account
- Selecting Events for a List of Accounts
- Selecting Events for a Sharing Group
- Selecting Events Associated with Offers
- Selecting Events Based On Event Type
- Selecting Events Based On Service Type
- Selecting Events Based On Subscriber Service ID
- Selecting Events Associated with Balance Groups
- Selecting Events Based on a Custom Event Field

Selecting Events for a Single Account

To select all events created by a single account after a specified date, you use the following syntax:

```
pin_rerate -a accountID -t timestamp
```

where:

- *accountID* is the account POID object ID. For example, if the account POID is **0.0.0.1 /account 8606 0**, the POID object ID is **8606**.
- *timestamp* is the timestamp in [*ss*/*mm*/*hh*/]*MM*/*DD*/*YYYY* format.

For example, this command finds all events generated by **0.0.0.1 /account 8606 0** after 23 July 2023 and adds them to a rerate job:

```
pin_rerate -a 8606 -t 07/23/2023
```

Alternatively, you can run this command in calculate-only mode by adding the **-c** parameter. It performs rerating but does not update anything in the BRM database. For example, to run the previous command in calculate-only mode, you would enter:

```
pin_rerate -a 8606 -c -t 07/23/2007
```

> **Note:**
>
> You can use the calculate-only mode (**-c** parameter) with only the **-a** parameter.

Selecting Events for a List of Accounts

To select all events created by a list of accounts after a specified date, you use the following syntax:

```
pin_rerate -m inputFile -t timestamp
```

where *inputFile* is the name of the XML file that lists the accounts.

For example, this command finds all events generated by the accounts listed in **accounts_to_rerate.xml** after 15 July 2023 and adds the events to a rerate job:

```
pin_rerate -m accounts_to_rerate.txt -t 07/15/2023
```

In the input file, specify each account in a results array that lists account POIDs. For example:

```
0 PIN_FLD_RESULTS                   ARRAY [1]
    1 PIN_FLD_ACCOUNT_OBJ            POID [0] 0.0.0.1 /account 12345 0
0 PIN_FLD_RESULTS                   ARRAY [2]
    1 PIN_FLD_ACCOUNT_OBJ            POID [0] 0.0.0.1 /account 12333 0
...
```

Selecting Events for a Sharing Group

To select all events created after a specified date for all accounts in a specified sharing group, you use the following syntax:

```
pin_rerate -G groupOwnerId -t timestamp
```

where *groupOwnerId* is the account ID of the sharing group owner.

The utility includes events for the owner account and all member accounts of the specified charge sharing group, discount sharing group, profile sharing group, or product sharing group in the rerate job.

> **Note:**
>
> This job option does not support multilevel sharing group hierarchies. Only the owner and their immediate children are included in the job.

By default, the owner and all members of the sharing group must reside in the same database schema. To rerate sharing groups with accounts in multiple schemas, you must set the **CrossSchemaSharingGroup** business parameter. See "Enabling Sharing Groups to Support Multiple Schemas" in *BRM Managing Customers* for more information.

For example, this command finds all events generated after 23 July 2023 by the sharing group owned by **0.0.0.1 /account 423591** and adds them to a rerate job:

```
pin_rerate -G 423591 -t 07/23/2023
```

Selecting Events Associated with Offers

You can select all events for accounts associated with one of the following:

- **A list of charge offers.** To do so, use the **-p** *inputFile* parameter to include all events for accounts associated with one or more specified charge offers. For example:

  ```
  pin_rerate -p charge_offer_rerate.txt -t 01/23/2023
  ```

- **A list of discount offers.** To do so, use the **-i** *inputFile* parameter to include all events for accounts associated with one or more specified discount offers. For example:

  ```
  pin_rerate -i discount_offer_rerate.txt -t 01/23/2023
  ```

- **A list of bundles.** To do so, use the **-d** *inputFile* parameter to include all events for accounts associated with charge offers or discount offers that belong to one or more specified bundles. For example:

  ```
  pin_rerate -d bundle_rerate.txt -t 01/23/2023
  ```

  If a charge offer or discount offer is part of multiple bundles, all accounts that purchase the charge offer or discount offer are selected for rerating even if they did not purchase the bundle that was rerated.

> **Note:**
>
> The **-p**, **-i**, and **-d** parameters are mutually exclusive.

In the input file, list the charge offer, discount offer, or bundle names with one name on each line. For example:

```
ChargeOffer_1
ChargeOffer_2
ChargeOffer_3
```

All names are case-sensitive and must match the names as defined in your Pricing Design Center (PDC) product offerings.

Selecting Events Based On Event Type

To select events created after a specified date for accounts associated with a specific event type, you use the following syntax:

pin_rerate -n *inputFile* **-t** *timestamp*

For example, this command finds all events created after 10 January 2023 for accounts that are associated with the event types listed in **event_rerate.xml**:

```
pin_rerate -n event_rerate.txt -t 01/10/2023
```

In the input file, you list the event types with one on each line. For example, this specifies to select only GSM and GPRS events:

```
/event/session/telco/gsm
/event/session/telco/gprs
```

When rerating cycle fee events, to get the correct rerating results, include the cycle events that occur during billing that are configured in the product, such as cycle discount, rollover, and fold events in the event file. For example, if a cycle discount is configured to be some percentage of the charge during billing and if the cycle forward arrears fee is modified during the billing cycle, then to rerate the cycle forward arrears event, you need to include both events in the event file:

- **/event/billing/product/fee/cycle/cycle_forward_arrear**
- **/event/billing/cycle/discount**

You can specify to also include the subclasses of all event types listed in the input file. To do so, include the **-r** parameter at the command line. For example:

```
pin_rerate -n event_rerate.txt -r -t 01/10/2023
```

In this example, the utility would also include the **/event/session/telco/gsm/data**, **/event/session/telco/gsm/voice**, and **/event/session/telco/gprs/session** events in the rerate job.

Selecting Events Based On Service Type

To select all events for accounts associated with one or more service types, you use the following syntax:

```
pin_rerate -s inputFile -t timestamp
```

For example, this command finds all events for accounts associated with the service types listed in **service_rerate.xml** created after 20 May 2023, and adds the events to a rerate job:

```
pin_rerate -s service_rerate.txt -t 05/20/2023
```

In the input file, you list the service types with one on each line. For example:

```
/service/telco/gsm/data
/service/telco/gsm/telephony
/service/telco/gsm/sms
```

Selecting Events Based On Subscriber Service ID

To select all events for accounts associated with a service identifier, you use the following syntax:

```
pin_rerate -line subscriptionID -t timestamp
```

where *subscriptionID* is the service identifier the customer created for the account. The service identifier is a unique ID that connects the customer to the service instance that the customer owns. For example, it could be a phone number, email address, or a personal ID number. Customers use the service identifier as their login name for the service.

For example, this command finds all events for accounts with the phone number 6495832245 created after 20 June 2023, and adds the events to a rerate job:

```
pin_rerate -line 6495832245 -t 06/20/2023
```

You can further restrict the selected accounts to those associated with:

- Charge offers (**-p**)
- Discount offers (**-i**)
- Bundles (**-d**)
- Events (**-n**)
- Services (**-s**)
- Accounts (**-m**)
- Custom fields (**-***field_name*)

Selecting Events Associated with Balance Groups

To select all events created for one or more balance groups, you use the following syntax:

```
pin_rerate -g inputFile -t timestamp
```

For example, this command finds all events generated by the balance groups listed in **balance_groups.xml** after 1 January 2023, and adds the events to a rerate job:

```
pin_rerate -g balance_groups.txt -t 01/23/2023
```

In the input file, provide a results array that lists the POIDS for accounts, bill units, and balance groups. To rerate events for all balance groups in an account's bill unit, specify a value of NULL as the POID ID for the balance group, as shown in the following example:

```
0 PIN_FLD_RESULTS                  ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ         POID [0] $DB_NO /account 12345 0
    1   PIN_FLD_BILLINFO_OBJ        POID [0] $DB_NO /billinfo 34567 0
    1   PIN_FLD_BAL_GRP_OBJ         POID [0] $DB_NO /balance_group 66765 0
0 PIN_FLD_RESULTS                  ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ         POID [0] $DB_NO /account 12344 0
    1   PIN_FLD_BILLINFO_OBJ        POID [0] $DB_NO /billinfo 45654 0
    1   PIN_FLD_BAL_GRP_OBJ         POID [0] $DB_NO /balance_group NULL 0
...
```

Selecting Events Based on a Custom Event Field

To select events for accounts associated with a custom event field, you use the following syntax:

```
pin_rerate -fieldName inputFile -t timestamp
```

where *fieldName* is the name of a field you define. For example, if tax was calculated incorrectly, you might define the custom parameter **-tax_supplier** for selecting events based on the PIN_FLD_TAX_SUPPLIER field. In this case, you would specify a list of tax supplier IDs in the input file.

When you use a custom event field, the utility searches the base **/event** storable class for the field.

If the custom event field is present only in a subclass, use the **-n** parameter instead. For example, the PIN_FLD_ORIGIN_NETWORK field is not in the base **/event** class, so you must use the **-n** parameter to select events based on that field. In this case, you create an input file specifying the **/event/activity/telco** event in a file named **event_rerate.txt**.

> **Note:**
>
> When using the **-n** parameter with a custom field, the input file can contain only one type of event.

To select accounts whose telco events were generated at the specified origin network, run the following command:

```
pin_rerate -n event_rerate.txt -origin origin.txt -t 06/01/2024
```

If the event specified in the input file contains subclasses, all subclass events are also selected. For example, if you specify **/event/activity/telco**, BRM also selects events from the **/event/activity/telco/gsm** event.

To define custom event fields:

1. Create an XML file that maps the event field name to the utility's parameter name according to the BRM rerating XML schema file (*BRM_home***/xsd/pin_rerate_flds.xsd**). You can map parameter names to fields in the base **/event** storable class or to fields in an **/event** subclass. For example:

```
<PinRerate>
   <Name>EVENT.PIN_FLD_TAX_SUPPLIER</Name>
   <value>tax_supplier</value>
</PinRerate>
```

where the **<Name>** tag specifies the event field name, and the **<value>** tag specifies the parameter name used at the command line.

> **Note:**
>
> Ensure that you validate your XML file against the XML schema. The **load_pin_rerate_flds** utility cannot load an invalid XML file.

2. Load the XML file into the **/config/rerate_flds** object by running the **load_pin_rerate_flds** utility:

   load_pin_rerate_flds -f *xml_file_name*

   See "load_pin_rerate_flds" for more information.

3. Run the **pin_rerate** utility with the *-fieldName* parameter:

   pin_rerate -*fieldName inputFile* **-t** *timestamp*

   For example, to search for the tax supplier in events:

   ```
   pin_rerate -tax_supplier tax_supplier.txt -t 01/23/2023
   ```

   The **pin_rerate** utility uses the **/config/rerate_flds** object to identify the event field to search for.

Backing Out Rating for Select Events

Backout-only rerating backs out the balance impacts of rating without rerating events. BRM backs out balance impacts for backout-only rerating by creating an adjustment event that entirely negates the original balance impacts.

> **Note:**
>
> Use caution when choosing the events to back out because it can impact your general ledger. For example, it is incorrect to use backout-only rerating for a cycle event when the customer has already paid the cycle fee or to use backout-only rerating when pricing is changed. Typically, backout-only rerating is performed only on usage events where rating should not have occurred.

To back out the balance impacts of rating, run **pin_rerate** with the **-backout** parameter. Use the **-backout** parameter with other parameters that select the accounts and their events to include in the job.

For example, the following command selects accounts whose events were rated by the charge offers specified in the **charge_offer_backout.txt** file and backs out those events that occurred after 12/1/2024:

```
pin_rerate -backout -p charge_offer_backout.txt -t 12/1/2024 -r
```

Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Events

> **Note:**
>
> - Rerating recurring events can affect performance.
> - Before you can rerate cycle forward and cycle forward arrears charges, you must enable this feature. See "Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Events".

You can create rerate jobs for cycle forward and cycle forward arrears events after their pricing has changed.

For example, suppose a $10 cycle fee is charged to an account for the cycle from April 15 to May 15. You change the fee from $10 to $20 on April 30, which is the middle of the cycle. When you rerate events, BRM recalculates the cycle fees as follows:

- Refunds the $10 for the old cycle fee.
- Recalculates the cycle fees based on the $10 fee for the first 14 days in the cycle and the $20 fee for the next 16 days in the cycle:

  ($10 x 14/30) + ($20 x 16/30) = $15.33

After you change the pricing for a cycle forward or cycle forward arrears event, run the **pin_rate_change** utility to analyze charge offers and create rerate jobs for cycle forward and cycle forward arrears events:

```
pin_rate_change
```

See "pin_rate_change" for more information.

Assigning Reason Codes to Rerate Jobs

You may sometimes need rerating jobs to be processed in a specific order, such as charge offers first, discount offers second, and bundles third. You can enforce the order in which jobs are processed by assigning a reason code to your rerate jobs. Later, you specify the order in which to process your rerate jobs with an assigned reason code when you run rerating. See "Specifying the Order to Process Rerate Jobs" for more information.

To assign a reason code to a rerate job, you include the **-reason** parameter when specifying the events to rerate. For example:

```
pin_rerate -reason reasonCode -p charge_offer_rerate.txt -t 01/30/2030
```

where *reasonCode* is a unique integer, except for the number **1**. The number **1** is reserved for internal use.

For example, to assign reason code **100** to a rerate job containing events associated with a single account, you could enter:

```
pin_rerate -reason 100 -a 86060 -t 01/30/2030
```

Getting an Estimated Number of Rerated Events

You can get an estimate of the number of events in a rerate job by including the **pin_rerate -e** parameter at the command line. You can use the **-e** parameter with any of the selection parameters, as shown below:

```
pin_rerate -e [-p|-i|-s|-d|-n|-fieldName] inputFile -t timestamp
```

You cannot use the **-e** parameter when selecting events associated with the following:

- A list of accounts (**-m**)
- Balance groups (**-g**)

When you run the utility with the **-e** parameter, it displays an estimated number of events that will be rerated on the screen without loading any data into the database. It also writes the estimate to the **pin_rerate.pinlog** file.

For example, to get an estimate of the number of events that will be rerated for a list of services, use this command:

```
pin_rerate -e -s service_rerate.txt -t 12/01/2024
```

Specifying the Event Sequence in a Rerate Job

Events in a job are rerated in sequence based on the event time, which is based on either:

- **The event occurrence time**: Events are rerated in the order the events ended. This is the default.
- **The time the event was created in the database**: Events are rerated in the order they were loaded into the BRM database.

The event time you specify might depend on how the original events were rated:

- Events that are rated or loaded into the BRM database as a result of offline charging can have a significant delay between the event end time and the time they are loaded into the database. Using the event end time reflects the real-time sequence of the original events. However, because batch events are recorded in the order they were created in the database, this makes it harder to predict the impact of a price configuration change. To compare the original and rerated balance impacts of batch events, use the event creation time.
- Events rated by online charging, and cycle events and purchase events, have no or very little delay between the event end time and the time they are loaded into the database. Both times reflect the real-time sequence in which the original events occurred and were recorded.

To specify the sequence in which to process events in a job, use the following syntax:

```
pin_rerate -b [c|e] eventCriteria
```

where:

- **-bc** specifies to rerate events in the order they were loaded into the BRM database, and **-be** specifies to rerate events in the order that the usage events occurred.
- **-be** specifies to rerate events in the order that the usage events occurred.
- *eventCriteria* is any event selection criteria, such as the **-a**, **-d**, **-g**, **-i**, **-m**, **-p**, **-s , -line**, or custom parameters.

For example, this command selects events for all accounts that occurred after 06/15/2024 and specifies to rerate in the order in which the events occurred.

```
pin_rerate -be -t 06/15/2024
```

5

Processing Events in Rerate Jobs

Learn how Oracle Communications Billing and Revenue Management (BRM) rerates the events in your jobs when you run the **pin_rerate** utility in processing mode.

Topics in this document:

- About the Rerating Process
- About Processing Rerate Jobs
- Specifying the Order to Process Rerate Jobs

About the Rerating Process

After a rerate job is created, BRM, ECE, and the **pin_rerate** utility work together to process the job as follows:

1. When you run **pin_rerate**, it finds all rerate jobs with a NEW status, sends a prepare-to-rerate message to ECE, and then updates the job's status to WAITING_ACCOUNT_LOCK.
2. ECE sends the rated events in the job to the BRM database to synchronize balances. When done, ECE sends an acknowledgment to **pin_rerate**.
3. The **pin_rerate** utility suspends rating for the accounts associated with the job and updates the job's status to ACCOUNT_LOCKED.

   BRM also begins storing subscription events that occur in the BRM system. After rerating, BRM rates the subscription events and sends any relevant results to ECE. For example, BRM sends balance updates and changes to the purchased offers.
4. The **pin_rerate** utility sends the events to rerate to ECE and updates the job's status to RERATED.
5. ECE rerates the events. ECE performs any necessary balance updates and backouts, and synchronizes the balances for accounts in the rerating requests.

   ECE moves any failed rerating requests to the Suspense queue.
6. ECE sends the charging results to the BRM database. When done, ECE sends an acknowledgment to the utility.
7. The **pin_rerate** utility resumes processing the accounts being rerated and sets the job's status to READY_TO_RECYCLE.
8. The **pin_rerate** utility recycles all events suspended during the rerating process and updates the job's status to COMPLETE.

> **Note:**
>
> If ECE cannot rerate events for an account, ECE sends a notification to BRM. In turn, BRM uses the information to create a new rerate job for the account.

During the rerating process, ECE continues rating usage and applying top-ups, but waits until rating completes to send the rated results to the BRM database.

About Processing Rerate Jobs

You process the rerate jobs in the queue by running the **pin_rerate** utility. You can use the utility to do the following:

- Rerate all new jobs in the queue. See "Steps for Processing Rerate Jobs".
- Rerate all new jobs in the queue and generate an output report. See "Generating Reports During Rerating".
- Recycle events suspended during the rerating process. See "Recycling Records Suspended During Rerating".
- Rerate any failed jobs in the queue. See "Processing Failed Account Events in Rerate Jobs".
- Purge completed and failed rerate jobs from the queue. See "Purging Rerate Jobs".

While running rerating, you can also specify the order in which to process multiple rerate jobs. See "Specifying the Order to Process Rerate Jobs".

Steps for Processing Rerate Jobs

You rerate the events in your rerate jobs by running the **pin_rerate** commands in the order shown below. You can run the commands manually or through a cron job.

> **Note:**
>
> Do not move accounts to another database schema while rerating events for those accounts.

To process rerate jobs in the queue, run these commands in order:

```
pin_rerate -process jobs
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

For a description of how the utility processes jobs, see "About the Rerating Process".

For more information about the utility's syntax and parameters, see "pin_rerate".

Generating Reports During Rerating

You can specify to generate a report during the rerating process. To do so, include the **-l** parameter when you run the **pin_rerate -rerate** command:

```
pin_rerate -rerate -l[d|s|sd|n]
```

When you include the **-l** parameter, the utility generates both a summary report (**pin_rerate.summary**) and a detailed report (**pin_rerate.detail**). You can specify the type of report to create by doing the following:

- To generate only a detailed report, use the -**ld** parameter:

  ```
  pin_rerate -rerate -ld
  ```

- To generate only a summary report, use the **-ls** parameter:

  ```
  pin_rerate -rerate -ls
  ```

- To generate both a detailed report and a summary report, use the **-lsd** parameter:

  ```
  pin_rerate -rerate -lsd
  ```

- To not generate any report, use the **-ln** parameter:

  ```
  pin_rerate -rerate -ln
  ```

The following shows the series of commands to run for processing rerate jobs and generating summary and detailed reports:

```
pin_rerate -process jobs
pin_rerate -process queue
pin_rerate -rerate -l
pin_rerate -process queue
pin_rerate -process recycle
```

Recycling Records Suspended During Rerating

Temporarily suspended records are loaded into the BRM database by the Suspended Event (SE) Loader. The suspended records are stored in the database until they are recycled.

> **Note:**
>
> Before you can recycle suspended records, they must be loaded into the BRM database by SE Loader. You typically schedule SE Loader to run automatically when you set up standard recycling.

To recycle the records suspended during the rerating process, run **pin_rerate** with the **-process recycle** parameter.

> **Note:**
>
> Suspended records are typically recycled by running the **pin_recycle** utility. However, **pin_rerate** calls the standard recycling opcode directly so you do not use **pin_recycle** when using **pin_rerate**.

The **pin_rerate** utility finds all rerate jobs with a status of READY_FOR_RECYCLE and calls the standard recycling opcodes to recycle the associated records. Standard recycling uses the recycle key value in the record to identify and retrieve records suspended during the rerating process.

After the records are recycled, **pin_rerate** changes the status of the jobs from READY_FOR_RECYCLE to COMPLETE.

> **Note:**
>
> - In a multischema environment, you must run **pin_rerate** separately on each database schema.
> - If no records were suspended for the accounts being rerated, the job's status is still changed to COMPLETE.
> - If an error occurs while recycling records, the job's status is not changed. It retains the status of READY_FOR_RECYCLE.

Processing Failed Account Events in Rerate Jobs

If rerating fails for one or more accounts in a job, the **pin_rerate** utility automatically creates a report that includes the account numbers and start times for each failure. The report file is named **pin_rerate.status_report** and is saved in the directory from where you ran the utility. When the rerating process is complete for the job, **pin_rerate** creates a new rerate job consisting of events for only the accounts that failed.

To process rerate jobs for accounts that failed, you run the same commands for processing any rerate jobs:

```
pin_rerate -process jobs
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

> **Note:**
>
> This processes all rerate jobs in the queue, and not just failed rerate jobs.

Purging Rerate Jobs

To purge all rerate jobs with a status of COMPLETE or UNSUCCESSFUL from the queue, you use the following syntax:

```
pin_rerate -purge
```

You can purge rerate jobs that occurred before a specified date and time by also including the **-t** parameter:

```
pin_rerate -purge -t timestamp
```

where *timestamp* is the timestamp in [*ss*|*mm*|*hh*|]*MM*|*DD*|*YYYY* format.

For example, this command purges all processed and failed rerate jobs created before 1 May 2023:

```
pin_rerate -purge -t 05/01/2023
```

Specifying the Order to Process Rerate Jobs

You may sometimes need to process rerating jobs in a specific order. For example, a job including events that impact future rating, such as quantity-based discounting, must be rerated first.

To specify the order in which to process rerate jobs:

1. Assign reason codes to your rerate jobs.
   - To assign reason codes when manually creating rerate jobs, see "Assigning Reason Codes to Rerate Jobs".
   - To assign reason codes during automatic job creation, see "Assigning Reason Codes During Automatic Job Creation".
2. Run the **pin_rerate** utility multiple times, assigning a reason code each time.

To process rerate jobs according to reason codes in a particular order, you must run rerating separately for each reason code. For example, to process a rerate job with reason code 100 first, a job with reason code 101 second, and a job with reason code 102 third, you would do the following:

1. Run the rerating process for the rerate job with reason code 100:

```
pin_rerate -process jobs -reason 100
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

2. Run the rerating process for the rerate job with reason code 101:

```
pin_rerate -process jobs -reason 101
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

3. Run the rerating process for the rerate job with reason code 102:

```
pin_rerate -process jobs -reason 102
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

You can optionally list multiple reason codes at the command line, but the utility will process the jobs randomly. They are not processed in the order you list them at the command line, nor are they processed in increasing or decreasing numeric order. For example, if you run the following command, the utility would process the three jobs in random order:

```
pin_rerate -process jobs -reason 103,104,105
```

6

Tuning Rerating Performance

Learn how to improve performance for Oracle Communications Billing and Revenue Management (BRM) rerating.

Topics in this document:

- Improving pin_rerate Performance
- Setting the Rerating Event Cache Size (Fetch Size)
- Configuring the Number of Accounts Per Job and Number of Jobs per Transaction

See also:

- About Rerating Events
- Configuring BRM to Create Rerate Jobs Automatically
- Processing Events in Rerate Jobs
- Configuring BRM to Rerate Events

Improving pin_rerate Performance

The following **pin_rerate** configuration parameters can be used with **pin_rerate -rerate** option to improve **pin_rerate** performance:

```
- pin_rerate  rerate_children  5
- pin_rerate  rerate_per_step  1000
- pin_rerate  rerate_fetch_size  5000
```

> **Note:**
>
> These entries can only be used with **pin_rerate -rerate** option.

Setting the Rerating Event Cache Size (Fetch Size)

By default, BRM rerating caches 10,000 events in system memory for processing. Depending on your system memory, you can set the **EventFetchSize** business parameter to specify the number of events retrieved from the database and cached in the system memory for processing.

To set the event cache size:

1. Go to *BRM_home*/**sys/data/config**.
2. 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 an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.

3. In **bus_params_subscription.xml.out**, set **EventFetchSize** to the required cache size:

   ```
   <EventFetchSize>value</EventFetchSize>
   ```

   The default value is **10000**.

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

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.
8. Stop and restart the CM.

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

Configuring the Number of Accounts Per Job and Number of Jobs per Transaction

To prepare for rerating, BRM assigns accounts to rerate jobs. By default, BRM assigns 10 accounts to each rerate job and creates 2 rerate jobs per transaction. For performance tuning, you can change the default number of accounts per job and the number of rerate jobs created per transaction:

1. Open the **pin_rerate** configuration file (*BRM_home*/**apps/pin_rerate/pin.conf**).
2. Set the number of accounts assigned to each rerate job by adding the following line:

   ```
   - pin_rerate per_job accounts_per_job
   ```

   where *accounts_per_job* is the number of accounts to assign to each job.

3. Set the number of jobs created per transaction by adding the following line:

   ```
   - pin_rerate per_transaction jobs_per_transaction
   ```

where *jobs_per_transaction* is the number of jobs to create in each transaction.

4. Save and close the file.

> **Note:**
>
> Setting the **pin_rerate per_job** entry to a small number, for example 1, results in many rerate jobs being created. Too many rerate jobs can affect your system's performance due to the rerate steps performed for each rerate job. Processing multiple accounts in one rerate job reduces the total number of rerate steps performed compared to processing those same accounts in multiple rerate jobs.

7

Rerating Utilities

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

Topics in this document:

- load_pin_rerate_flds
- pin_rate_change
- pin_rerate

load_pin_rerate_flds

Use this utility to load the following information into the BRM database:

- Custom event selection parameters. The data is loaded into the **/config/rerate_flds** object. See "Selecting Events Based on a Custom Event Field".
- Balance-impact comparison fields that determine if an adjustment event must be created. The data is loaded into the **/config/rerate_flds/compare_bi** object. See "Configuring Rerating Adjustment Events".

Location

BRM_home/**sys/data/config**

Run **load_pin_rerate_flds** from this directory.

Syntax

```
load_pin_rerate_flds -f xml_file_name [-v] [-h]
```

Parameters

-f *xml_file_name*
Specifies the name of the XML file that contains the data.

> **Note:**
>
> The file you load cannot include both event selection data and balance-impact comparison data. You must load the files for these configurations separately.

-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:
> **load_pin_rerate_flds** *any_other_parameter* **–v >** *filename***.log**

-help
Displays the syntax and parameters for this utility.

Results

This utility notifies you when it successfully creates the **/config/rerate_flds** object.

If it cannot create the **/config/rerate_flds**, it logs an error in the log file (**default.pinlog**). It creates the log file either in the directory from which the utility was started or in the directory specified in the configuration file.

pin_rate_change

Use this utility to create rerate jobs for cycle forward and cycle forward arrears events when the pricing changes. Use this BRM utility after you change pricing in PDC and before running the **pin_rerate** utility.

See "Creating Rerate Jobs for Cycle Forward and Cycle Forward Arrears Charges".

The **pin.conf** file for this utility is in *BRM_home***/apps/pin_rate_change**.

Location

*BRM_home***/bin**

Syntax

```
pin_rate_change [-v] [-d] [-h]
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter 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_rate_change** any_other_parameter **–v >** filename**.log**

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-h
Displays the syntax and parameters for this utility.

Results

The **pin_rate_change** utility notifies you when it successfully creates the rerate jobs

If the **pin_rate_change** utility does not notify you that it was successful, run the command again with the **-d** parameter, and 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_rerate

Use this BRM utility to rerate events. See "About Rerating Events".

To rerate events, you run the **pin_rerate** utility multiple times: once to create rerating jobs and several additional times to process the rerate jobs.

> **Note:**
>
> - **pin_rerate** is a multithreaded application (MTA).
> - In a multischema environment, you must run **pin_rerate** separately on each database schema.

Location

*BRM_home***/bin**

The **pin.conf** file for this utility is located in *BRM_home***/apps/pin_rerate**. Run **pin_rerate** from this directory.

Syntax Overview

You can run the **pin_rerate** utility in the following modes:

- Select mode: Select events for rerating and bundles them into a rerate job. See "Rerate Job Creation Options".
- Process mode: Processes the rerate jobs. See "Rerate Job Processing Parameters".
- Configuration mode: Provides additional options when rerating jobs. See "Rerate Job Configuration Options".

Results

If the **pin_rerate** utility does not notify you that it was successful, look in the utility log file (**pin_rerate.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.

If rerating fails, the utility creates a report that includes the account numbers and start times for failed rerates. The report file name is **pin_rerate.status_report**, and is in the directory from where you ran the utility.

Rerate Job Creation Options

When you run the **pin_rerate** utility with the job creation options, it finds the events to rerate and bundles them into a rerate job. See "About Manually Selecting Events for Rerate Jobs".

Syntax for Creating Rerate Jobs

```
pin_rerate  -a accountPOID  [-c]
            -t timestamp
            -G groupOwnerPoid -t timestamp
            -p|-i|-s|-d|-n|-m|-g|-fieldName inputFile -t timestamp [-r] [-e]
            -line subscriptionID -t timestamp
            [-reason reason_code]
```

Parameters for Selecting Accounts for Rerating

All account selection parameters are optional and mutually exclusive, except for the **-t** parameter, which is mandatory when selecting the events to rerate. If you specify only the time (**-t**) parameter, BRM selects all accounts for rerating in the period defined by the **-t** parameter.

-a *accountPOID* [-c]
Includes the events associated with the specified account POID in the rerate job.
You can also include the **-c** option to rerate in calculate only mode. The utility rerates the events without updating the database.

-t *timestamp*
Selects accounts for rerating based on the event start time. All accounts with events that occurred between the start time and the current date are included in the rerate job.

-G *groupOwnerPoid* -t *timestamp*
Selects accounts for rerating based on the specified sharing group owner. The owner account and all member accounts of the specified charge sharing group, discount sharing group, profile sharing group, or product sharing group are included in the rerate job.
All accounts with events that occurred between the start time and the current date are included in the rerate job.
By default, the owner and all members of the sharing group must reside in the same database schema. To rerate sharing groups with accounts in multiple schemas, you must set the **CrossSchemaSharingGroup** business parameter. See "Enabling Sharing Groups to Support Multiple Schemas" in *BRM Managing Customers* for more information.

> **Note:**
>
> This job option does not support multilevel sharing group hierarchies. Only the owner and its immediate children are included in the job.

-p *inputFile* -t *timestamp*
Selects accounts for rerating based on the charge offers listed in the input file. Accounts that have events associated with the charge offers specified in *inputFile* are included in the rerate job.
In the input file, include a line for each charge offer name. All names are case sensitive.

-i *inputFile* -t *timestamp*
Selects accounts for rerating based on the discount offers listed in the input file. Accounts that own at least one instance of the discount offer specified in *inputFile* are included in the rerate job.
In the input file, include a line for each discount offer name. All names are case sensitive.

-s *inputFile* -t *timestamp*
Selects accounts for rerating based on the service names listed in the input file. Accounts that have events associated with the services specified in *inputFile* are included in the rerate job.
In the input file, include a line for each service name. All names are case sensitive.

-d *inputFile* -t *timestamp*
Includes in the rerate job all events associated with the bundles listed in the input file. Accounts that have events associated with the charge offers and discount offers that belong to the bundles specified in *inputFile* are selected for rerating.
In the input file, include a line for each bundle name. Names are case sensitive.

> **Note:**
>
> If a charge offer is a part of multiple bundles, all accounts that purchase the charge offer are selected for rerating even if they did not purchase the bundle that was rerated.

-n *inputFile* -t *timestamp*
Selects accounts for rerating based on the types of events. Accounts that have events of the types specified in *inputFile* are selected for rerating.
In the input file, include a line for each bundle name. Names are case sensitive.

> **Note:**
>
> - If you use this parameter with the **-r** parameter, all subclasses of the event specified in the input file are also rerated. See "Selecting Events for Rerate Jobs".
> - When a custom **pin_rerate** parameter is used, the **-n** parameter is mandatory if the custom parameter is based on an event field present only in an event subclass. In this case, the **-n** parameter input file can contain only one type of event.

-m *inputFile* -t *timestamp*
Selects a set of accounts for rerating based on the provided account POIDs.
inputFile format: A text file containing the accounts to rerate in flist format. Specify each account in a results array. For example:

```
0 PIN_FLD_RESULTS                  ARRAY [1]
    1 PIN_FLD_ACCOUNT_OBJ           POID [0] $DB_NO /account 12345 0
0 PIN_FLD_RESULTS                  ARRAY [2]
    1 PIN_FLD_ACCOUNT_OBJ           POID [0] $DB_NO /account 12333 0
...
```

where PIN_FLD_ACCOUNT_OBJ is the POID of the account object.

You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-n**, **-p**, or **-s -e**, **-line**, or *field_name* parameters.

-g *inputFile* -t *timestamp*
Selects accounts for rerating based on balance groups.
inputFile format: A text file containing information for one or more accounts in flist format. Specify an account POID, bill unit POID, and balance group POID in a results array for each account. For example:

```
0 PIN_FLD_RESULTS                 ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ          POID [0] $DB_NO /account 12345 0
    1   PIN_FLD_BILLINFO_OBJ         POID [0] $DB_NO /billinfo 34567 0
    1   PIN_FLD_BAL_GRP_OBJ          POID [0] $DB_NO /balance_group 66765 0
0 PIN_FLD_RESULTS                 ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ          POID [0] $DB_NO /account 12344 0
    1   PIN_FLD_BILLINFO_OBJ         POID [0] $DB_NO /billinfo 45654 0
    1   PIN_FLD_BAL_GRP_OBJ          POID [0] $DB_NO /balance_group NULL 0
...
```

To rerate all events for all balance groups for a given account's bill unit, specify a value of NULL as the POID ID for the balance group object, as shown in the second results array in the above example.

-fieldName inputFile* -t *timestamp
field_name is a custom **pin_rerate** parameter you have defined that maps to a specific event field.
inputFile is a file that contains the values of the event field used to select accounts for rerating.
Selects accounts based on the custom parameter. Accounts are selected for rerating that have events with the field identified by field_name whose field value is one of those specified in *inputFile*.
For information about defining custom **pin_rerate** parameters, see "Selecting Events Based on a Custom Event Field".

> **Note:**
>
> If the custom parameter maps to a field in an ***/event*** subclass, you must specify the subclass event by including the **-n** parameter, and the **-n** parameter input file can include only one event. If you specify more than one type of event, an error is returned.

-line *subscriptionID*
Selects accounts for rerating based on a subscription service. The subscription service is identified by the PIN_FLD_ALIAS_LIST.PIN_FLD_NAME field, which can specify, for example, the caller ID such as a phone number or the MAC address.

-reason *reasonCode*
Use with other selection parameters to assign a rerate reason code to the jobs created for the selected accounts.
For more information about assigning rerate reason codes, see "Specifying the Order to Process Rerate Jobs".

-r
Specifies to rerate all subclasses of the event specified in the input file.
You can use the **-e** parameter with the **-p**, **-i**, **-s**, **-d**, **-n**, **-m**, **-g**, and **-*fieldName*** parameters.

-e
Displays an estimated number of events that will be rerated to the screen without loading any data into the database. It also writes the estimate to the **pin_rerate.pinlog** file.
You can use the **-e** parameter with the **-p**, **-i**, **-s**, **-d**, **-n**, and **-***fieldName* parameters.

Rerate Job Processing Parameters

When you run the **pin_rerate** utility with the job processing parameters, it processes the rerate jobs in the queue. For more information, see "Steps for Processing Rerate Jobs".

Syntax for Processing Rerate Jobs

```
pin_rerate    -process jobs [-reason reasonCode]
              -process queue
              -process recycle [-db databaseID]
              -rerate [-l [d | s | sd | n]]
              -purge [-time timestamp]
```

All parameters are optional and mutually exclusive.

Parameters for Processing Rerate Jobs

-process jobs [-reason *reasonCode*]
Searches for all jobs with a NEW status, sends the list of accounts associated with the job to ECE to lock the accounts, and then sets the job's status to WAITING_ACCOUNT_LOCK.
Include the optional **-reason** *reasonCode* option to process only jobs associated with the specified reason codes. Replace *reasonCode* with a comma-separated list of reason codes.
See "Specifying the Order to Process Rerate Jobs" for more information.

-process queue
Processes acknowledgment events from ECE and then updates the job's status to ACCOUNT_LOCKED.
When processing rerate jobs, you run **pin_rerate -process queue** twice: once before running utility with the **-rerate** parameter and again after running the utility with the **-rerate** parameter.

-process recycle [-db *databaseID*]
Recycles event records that were suspended during the rerating process and then changes the job's status to COMPLETED. By default, it recycles event records from the current database schema.
To recycle suspended event records residing in a different schema, include the **-db** *databaseID* option with the database number (for example, 0.0.0.2).

-rerate -l [d | s | sd | n]
Rerates the events in the job, releases the lock on all accounts associated with the job, and then changes the job's status to RERATED.
If you include the **-l** parameter, it also generates a detailed rerating report and summary rerating report. To change which reports are generated, use the following:

- **-ld**: Generates a detailed rerating report (**pin_rerate.detail**) only.
- **-ls**: Generates a summary rerating report (**pin_rerate.summary**) only.
- **-lsd**: Generates both the detailed and summary rerating reports. This is the default.
- **-ln**: Generates no report.

-purge [-t *timestamp*]
Purges all rerate jobs with a status of COMPLETED or UNSUCCESSFUL. Also include the **-t** *timestamp* option to purge completed and unsuccessful jobs created before the specified time in [*ss*|*mm*|*hh*|]*MM*|*DD*|*YYYY* format.

Rerate Job Configuration Options

When you run the **pin_rerate** utility with the job processing parameters, it processes rerate jobs. For more information, see "Steps for Processing Rerate Jobs".

Syntax for Processing Rerate Jobs

```
pin_rerate    [-b [c | e]] [-e  -a | -s | -p | -n | -d] [-backout] [-help]
```

Parameters Affecting Rerating Behavior

The following are additional, optional parameters that affect rerating behavior and the utility's output.

-b [c | e]
Specifies the order in which *batch* events are rerated based on the event time. You can specify one of two times:

- **-bc**: Rerates events in the order that they were loaded into the BRM database.
- **-be**: Rerates events based on the time the event occurred. This is the default.

See "Specifying the Event Sequence in a Rerate Job" for more information.

-e -a | -s | -p | -n | -d
Returns an estimate of how many events might be affected by rerating based on which accounts are being rerated:

- **-e -a**: Provides an estimate for a single account.
- **-e -s**: Provides an estimate for accounts associated with a specific service.
- **-e -p**: Provides an estimate for accounts that own with specific charge offers.
- **-e -n**: Provides an estimate for accounts with specific types of events.
- **-e -d**: Provides an estimate for accounts that own specific bundles.

If you do not specify one of these options when using the **-e** parameter, the utility returns **0**.

> **Note:**
>
> You cannot use this parameter with the **-m** or **-g** parameters.

-backout
Specifies backout-only rerating, which backs out the balance impacts of rating without rerating events. For more information about back-out-only rerating, see "Processing Events in Rerate Jobs".
This parameter is optional and can be used with any other parameter.

> **Note:**
>
> When choosing the events to back out, do not select events that could imbalance your general ledger, such as events for which fees have already been paid and usage events that should be rerated. Typically, backout-only rerating is performed only on usage events where rating should not have occurred.

-h | -help
Displays the syntax and parameters for this utility.